Visual Control of Legged Robots for Locomotion in Complex Environments using Central Pattern Generators

THÈSE N^{O} 6047 (2014)

PRÉSENTÉE LE 24 JANVIER 2014

À L'ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

À LA FACULTÉ DES SCIENCES ET TECHNIQUES DE L'INGÉNIEUR

LABORATOIRE DE BIOROBOTIQUE

ET

À L'INSTITUTO SUPERIOR TÉCNICO (IST) DA UNIVERSIDADE DE LISBOA

INSTITUTO DE SISTEMAS E ROBOTICA

PROGRAMME DOCTORAL EN SYSTÈMES DE PRODUCTION ET ROBOTIQUE

ET

DOUTORAMENTO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES (PhD)

PAR

Sébastien GAY

acceptée sur proposition du jury:

Prof. A. Martinoli, président du jury
Prof. A. Ijspeert, Prof. J. Santos-Victor, directeurs de thèse
Prof. K. Aminian, rapporteur
Prof. A. Bernardino, rapporteur
Prof. A. Lewis, rapporteur

Suisse
2014

Acknowledgements

First and foremost I owe particular thanks to my advisor Pr. Auke Ijspeert, for giving me the opportunity to work in his team. I believe the success of research work lies in finding the right balance between guidance and trust, realism and optimism, conformism and eccentricity, rigor and dream. Auke has found this perfect balance and I am glad he was able to transfer some of it to me during these four years. He has been both an encouraging supervisor and a great friend.
Next I would like to thank Pr. José Santos-Victor for his insightful advice, especially during my stay in Portugal.
I am also grateful to all the members of BioRob for creating such a nice atmosphere, whether for working or having fun. I have come to realize that one common cause of failure for PhD candidates is depression and loss of motivation, which seems simply not possible at BioRob.
The same goes for the VisLab team at IST in Lisbon, who made my stay in Portugal both a great working and human experience.
I would like to thank my family and especially the tremendous work done by my parents to support me through the carelessness of childhood and the laziness of adolescence.
A warm thank you to Sandra for staying by my side and bringing me so much during these four years while still somehow putting up with me.
Next come all my friends who I will not name by fear of forgetting some, but who have made my life in Lausanne and Lisbon fantastic.
Last but not least I would like to acknowledge the work done by Kuching, my cat, for always keeping my lap warm and providing me with a constant supply of mice.

This work was funded by the IST-EPFL initiative and the Fundação para a Ciência e a Tecnologia (FCT).

Lausanne, *Sébastien Gay*
8th January 2014

Abstract

THIS thesis proposes methods to use vision in the context of locomotion of legged robots on difficult terrain. Vision is often used only at a high-level for robot locomotion - for the segmentation of objects, the modeling of the surrounding environment etc. - while the lower levels of the controllers rely more on proprioceptive and vestibular sensors. In contrast, we propose to investigate the use of vision at various levels of the locomotion problem.

Here we specifically aim to address the following questions:
What type of visual information is relevant for which subproblem of locomotion ?
What needs to be modulated in the locomotion patterns to allow adaptation of the nominal gait to rough terrain ?
How to interface the visual system with the locomotor system ?
How to integrate information from the visual system and the other sensors ?
How can the robot motion be modulated to enhance its visual capabilities ?
How can vision help controlling a compliant robot ?

First, we use vision at a high level on top of a central pattern generator (CPG) controlling the locomotion of a humanoid robot crawling. Using vision to track surrounding objects, and artificial potential fields to plan the path, we enable the robot to crawl in a complex and dynamically changing environment, to reach predefined goals with its hand.

Then, we investigate how to use motion for the sake of vision. We propose an approach using adaptive frequency oscillators to stabilize the gaze of a legged robot while it is walking. Our approach contrasts from the state of the art by the fact that it only uses vision as sensory input to achieve gaze stabilization, while most existing methods use additional vestibular information. Thus it is applicable to very simple and cheap robots.

Finally we show that vision can be used at the lowest levels of robot motion control. We use vision, alone or fused with other sensors, for the control of balance. The controller is based on a CPG capable of performing discrete or periodic motions. A neural network is then used to learn the mapping between the sensory information and

modifications to the dynamics of this CPG. We show how this control framework can be used to command a humanoid standing on a randomly moving platform and to enable a quadruped robot to walk on uneven terrain.

Keywords: legged robots, vision, rough terrain locomotion, central pattern generators, dynamical systems, neural networks, sensory feedback.

Résumé

Cette thèse propose des méthodes pour utiliser la vision dans le contexte de la locomotion des robots à pattes sur terrain accidenté. La vision est souvent utilisée à un haut niveau pour la locomotion des robots - pour la segmentation d'objets, la modélisation de l'environnement immédiat etc. - alors que les niveaux les plus bas des contrôleurs utilisent plutôt des senseurs proprioceptifs et vestibulaires. Nous proposons au contraire d'étudier les utilisations de la vision à différents niveaux du problème de la locomotion.

Ici, nous visions spécifiquement à adresser les questions suivantes:
Quel type d'information visuelle est pertinente pour quel sous-problème de la locomotion ?
Comment les shémas de locomotion doivent être modulés pour permettre l'adaptation de la marche à des terrains accidentés ?
Comment interfacer le système visuel et le système de locomotion ?
Comment intégrer l'information du système visuel et des autres senseurs ?
Comment moduler les mouvements du robot pour améliorer ses capacités visuelles ?
Comment la vision peut aider à contrôler un robot élastique ?

Tout d'abord nous utilisons la vision à haut niveau avec un réseau locomoteur spinal (central pattern generator, CPG) qui contrôle la locomotion d'un robot humanoïde marchant à quatre pattes. En utilisant la vision pour suivre les objets environnants, des champs de potentiels artificiels pour planifier le chemin, nous permettons au robot de marcher dans un environnement complexe et changeant dynamiquement, afin d'atteindre des buts prédéfinis avec sa main.

Puis, nous étudions comment utiliser les mouvements du robot au profit de la vision. Nous proposons une approche utilisant des oscillateurs à fréquence adaptive (adaptive frequency oscillators, AFO), pour stabiliser le regard d'un robot pendant la marche. Notre approche contraste avec l'état de l'art dans le fait qu'elle n'utilise que la vision comme entrée sensorielle pour stabiliser le regard, alors que la plupart des méthodes existantes utilisent également l'information vestibulaire. Donc elle est applicable à des robots très simples et bon marché.

Finalement nous montrons que la vision peut être utilisée aux niveaux les plus bas du contrôle de la locomotion des robots. Nous utilisons la vision, seule ou fusionnée avec d'autres senseurs, pour le contrôle de l'équilibre. Le contrôleur est basé sur un CPG capable d'exécuter des mouvements discrets ou périodiques. Un réseau de neurones est alors utilisé pour apprendre les relations entre l'information sensorielle et les modifications de la dynamique de ce CPG. Nous montrons que ce système de contrôle peut être utilisé pour commander un humanoïde debout sur une plateforme qui bouge aléatoirement et pour permettre à un robot quadrupède marcher sur un terrain irrégulier.

Mots-clés: robots à pattes, vision, locomotion sur terrain accidenté, réseaux locomoteurs spinaux, systèmes dynamiques, réseaux de neurones, rétroaction sensorielle

Contents

List of Figures

List of Abbreviations

AFO	Adaptive Frequency Oscillator
ANN	artificial neural network
CPG	central pattern generator
COM	center of mass
COP	center of pressure
DOF	degree of freedom
IMU	inertial measurement unit
PID	proportional-integral-derivative
PSO	Particle Swarm Optimization
VMC	Virtual Model Control
ZMP	Zero Momentum Point

Chapter 1

Introduction

Vision is a very rich sensor, probably one of the richest available to both robots and animals in terms of the amount of information it delivers. It provides an accurate description of the surrounding environment, the position, colors, shapes and physical properties of objects, as well as their movements. Vision is critical to most animals for survival, since it signals the presence of predators and food sources. Vision also strongly helps for more self-centered tasks such as precise end-effector placement, balance control and ego-motion estimation. In robotics, vision has to this date mostly been used for high-level tasks such as path planning, object segmentation and manipulation. For lower level tasks especially in the context of legged locomotion, proprioceptive and vestibular information is usually preferred. I strongly believe that vision has not been exploited to its full potential, especially for legged robot locomotion control. This is the main motivation of this thesis.

The possible applications of vision in robotics are very wide and go much further than the framework of this thesis. I am taking interest here in how to use vision in the context of legged robots locomotion. Locomotion of legged robots is a difficult problem, and although impressive progress has been made in the recent years, the ability of robots to walk in changing environments remains much lower to that of animals. Adaptive locomotion control needs to tackle a number of aspects of robotics, from pattern generation to environment modeling, balance control, and dealing with compliance. I am deeply interested in this field of research and believe that vision can contribute to the control of legged locomotion more than it does to this point.

Although this thesis is meant to target mainly the robotics community, it is loosely inspired by biology both by the type of control it uses and the field of application. Throughout this thesis, vision is used for path planning, reaching, gaze stabilization, postural control and gait stabilization. As detailed in Section 1.1, these topics have been studied in animals and humans, and vision has proven to play a role in them.

After detailing research made in biology on the influence of vision in various aspects of animal and human locomotion, I will give a state of the art of the use of vision in the control of locomotion in robotics. I will also present a small review of research on central pattern generators (which we use at the low-level locomotion control), and optical flow (which we use as main visual information). Then I will explain the contribution of this thesis to the field, the questions addressed and the structure of the document. The state of the art presented in this chapter should be viewed as a general introduction to the field, while more detailed reviews about the specific problems tackled here can be found in the respective chapters.

1.1 Vision in Biology

In this section we review studies investigating the role of vision mainly for human locomotion. Please note that vision and more specifically optical flow has been studied extensively in insects, which use optical flow for navigation and landing (see [30] and [107] for a review). These studies are however outside of the scope of this review since we only address legged locomotion in this thesis.

While it is commonly accepted that vision plays important roles in reaching and grasping movements, until recently the studies of the influence of vision for locomotion has been almost only limited to the studies of anticipative strategies. Indeed, for humans and most mammals it is the only sensor capable of providing information information about the near to distant environment accurate enough to allow for predictive actions. Studies about the anticipative role of vision in the context of locomotion have been numerous and show various strategies used by humans to pass a narrow opening ([47]), step over obstacles ([72]), or avoid moving obstacles ([43]), for instance. While the strategies employed differ substantially depending on the task, observations common to all problems seem to confirm that human gaze is directed toward far space. Furthermore removing vision during the last step before reaching the obstacle or the target usually does not impair the performance. This proves that vision is used, at least in part, in an offline manner to plan future motion. For a good review about the predictive role of vision, see [46].

In the last decades however, vision has been proven to influence animal locomotion much more than was thought before.

The direction of gaze has shown to play an important role in the control of steering in human locomotion ([14]). When humans walk, the gaze direction moves towards the future heading direction prior to the body movements. Gaze actually leads the reorientation of the body ([23]).

In [62] and [66] the well-known moving room experiment showed that vision can in some cases override the vestibular system when controlling balance. The experiment consisted of having people stand in a room of which the walls and ceilings could be moved forward or backward and sideways. The floor was not moved so

the proprioceptive input did not change. However with the visual input conflicting with the proprioceptive information, subjects initiated unnecessary corrective postural adjustments when the walls were moved. In the case of babies, or adults standing on one foot, moving the room could cause falling. The subject would even give a verbal assessment that he was moving forward or backward, while he was in fact standing still. Similar results were obtained by moving the subject and the room together, while a non moving object was placed in his field of view. This time the subject would assess that the object is in fact moving even though his own motion was causing sway. This study shows that vision not only plays a role in balance control, which is critical for adaptive locomotion, but also can in some cases become the most important sensory cue.

In [81] Aftab E. Patla reviews the work done in his laboratory on visual control of locomotion in humans. When studying subjects stepping over an obstacle, he shows that removing vision for the last step before the obstacle does not impair the performance. This seems to suggests that exteroceptive vision (the part of the vision which provides information about the surrounding environment) is used at a high-level, offline, to detect the position of the obstacle and plan the motion ahead, while only proprioception is used during the actual stepping motion. However, removing peripheral vision during the whole run increased the toe clearance and had effect on the adjustments performed before reaching the obstacle. This shows that exproprioceptive vision (i.e. the part of the vision field providing information about ones own kinematics) is used to finely tune the trajectory of the leading limb for stepping over an obstacle. In another experiment Patla's team showed that optical flow is used to tune the position of the center of mass when walking. When gradually deteriorating the visual information, the authors could show that the amplitude of the COM deviation from its nominal trajectory increased.

In [71], proof is given that vision, more precisely optical flow, influences the speed of gait transition from walking to running. Subjects were placed on a treadmill running at gradually increasing speed. In front of the subject was a screen displaying forward motion in an endless corridor. The speed of this virtual forward motion could be controlled independently from the speed of the treadmill. The authors showed that gait transition from walking to running occurs at slower velocities when the speed of the visual (virtual) motion is faster than that of the treadmill, and at faster velocities in the inverse case. A further experiment, where the treadmill was this time controlled by the user showed that the preferred walking speed is influenced by the optical flow.

In many aspects of locomotion, optical flow is often identified as the part of vision being used the most (compared to stereoscopic depth estimation or specific object tracking for instance). In [74], the authors showed that optical flow is essential to accurately estimate the traveled distance. In contrast, they proved that other parts of the visual sense (depth cues and texture regularity) were not necessary for traveled distance estimation. In [115] the authors showed a strong correlation between the amount of optical flow available and the heading precision when humans were asked

to head towards a virtual target. Similarly, studies showed the central role of optical flow for balance control. In [12] different aspects of optical flow patterns (radial, intermediate and lamellar) are investigated in terms of their influence on the postural adjustments observed. By displaying specific optical flow patterns corresponding to non correlated self-motion, while the subjects were walking on a treadmill, the authors were able to quantify the postural adjustments of each specific pattern. Humans even seem to adopt special gaze behaviors to maximize the amount of optical flow produced and its accuracy ([90]).

As much as the study of vision and its influence on locomotion is interesting in itself, its relations with other sensors should not be neglected. Investigating the way the brains integrates different sensory cues is a relatively new field of research. In [10] trained Macaques were moved on a cart while viewing a virtual environment and asked to discriminate between left and right heading directions. The presented proprioceptive and visual inputs could thus be controlled independently. The authors showed that the error is decreased significantly when the two sensory cues are coherent. By actually measuring the activity of a particular group of neurons responsible for sensory integration, called the MSTd "congruent" neurons, they also showed that their response was enhanced when visual and vestibular cues concurred. Finally they proposed a Bayesian model for the sensory reweighing of multi-sensory integrating neurons. See for [73] a complete review on multi-sensory integration.

Summarizing the results of the above-mentioned biological studies, we can extract the following key points, essential as a basis for this thesis:

- Vision is essential to various aspects of locomotion.
- Vision is a complex sensor providing both exteroceptive and exproprioceptive information.
- Vision is used not only for planning but also for reactive actions.
- One of the most used part of vision for locomotion is optical flow.
- Vision and vestibular information are closely linked together, both at the behavioral and neural level.

In the next section I will give an overview of the use of vision for robotics control. This review is meant to be general and more problem specific descriptions of the state of the art may be found in the corresponding chapters (Chapter 2 to 6).

1.2 Vision in Robotics

Vision is used extensively in robotics, for a wide variety of tasks, including object tracking and segmentation, mapping, navigation, visual servoing, action recognition,

and more cognitive related applications like attention and affordances. This thesis studies vision in the framework of locomotion, placing it close to the areas of visual servoing and navigation. For a thorough review of vision in robotics, see [60].

Examples of using vision for path planning include autonomous cars with the Grand DARPA Challenge ([109]) where cars are required to drive in a natural environment without any human on board or remote controlling them. In [61] a rover is equipped with stereo vision cameras to establish a map of the surrounding environment while driving, and plan its path to find the easiest route. A more detailed review of path planning using vision is given in Chapter 2.

Vision is seldom used as an embedded sensor for legged robots, since the complex self-motion of the robot walking makes image processing, localization and mapping difficult. For the well known Little Dog Learning Locomotion project, vision was used extensively to estimate the geometry of the terrain to cross ([59], [84]). However, vision was external to the robot, which simplifies the problem since the cameras are not moving, and are perfectly positioned to have the whole terrain and the robot in the field of view. Moreover, the visual information was provided by multiple cameras (typically a VICON system), positioned all around the scene. This is much more detailed information that what can possibly be embedded in an autonomous mobile robot. In [58], the authors use embedded stereo vision on the Little Dog robot, enabling it to walk over the same kind of terrain as when using external cameras. The embedded cameras were placed on a stick to increase their height with respect to the robot, giving it a kind of giraffe look and allowing for longer range field of view. In [118] a similar footstep planner was implemented on the Ambler robot, a six-legged robot equipped with an embedded laser range-finder to estimate the terrain geometry. An example of vision based motion planning of a humanoid robot is described in [52]. The authors present an integrated framework where vision is used to plan the motion of a ZMP (*Zero Momentum Point*) controlled humanoid robot. Vision is used for path planning of the robot in its environment as well as to compute a collision free path for the movement of its arms. In [36], a quadruped robot is controlled by a central pattern generator integrating information from multiple sensors as sensory feedback. Vision is used to estimate the irregularities of the terrain and adapting the walking cyclic period and the contraction of the leg to stabilize the gait of the robot. Few approaches exist for the control of balance of legged robots using vision. In [76] and [80], visual information is used to estimate the position of the ZMP and achieve robot stability. Both approaches rely on the position of a reference object to estimate the pose of the robot.

Finally one of the most relevant example for this research is the work of Anthony Lewis on the coordination of vision and locomotion on simple central pattern generator driven biped robots. In [65] the depth information is extracted from images of two cameras and used to estimate the distance of an obstacle, and modulate the CPG parameters to adapt the step length and leg contraction to step over an obstacle. The

CPG modulation was done by a simple burst length neuron, which synaptic weight were adapted by a set of very simple rules. The robot was able not only to detect an obstacle during the last stride and step over it, but also adapt its stride length during the last few steps to make the crossing possible. A similar approach is described in [64] using optical flow instead of depth as a visual cue. This time a neural predictor is used to learn what should be the optical flow pattern when walking undisturbed. These patterns are then canceled out to detect anomalies and reinforcement learning is used to link these perturbations to reflexes in the CPG controlling the robot. Obstacles are detected this way and the learning process allows the robot to step over them.

This last piece of work is very relevant for Chapter 5 and 6, and a more information may be found in the respective literature reviews.

Note that one important field of research is the use of optical flow in flying robots. We do not review this field of research here as our topic is legged locomotion but a starting point for the interested reader should be to check the works of Mandyam Srinivasan ([106], [104]), Nicolas Franceschini ([35], [98]) and Dario Floreano ([120], [121]).

1.3 Optical flow algorithms

As mentioned in Section 1.1, optical flow is one of the most basic and yet one of the most useful information provided by the visual system in the context of legged locomotion. It contains a lot of information both about the environment and the motion of the individual. Flying insects like flies or bees typically use it for a variety of actions, like navigation, landing, self-motion estimation etc. ([29]). Optical flow can be used to compute the ego-motion of a robot, which is very relevant for this thesis. Thus it encapsulates information about the instantaneous stability of the robot. It also includes information about the shapes and distance of the surroundings, and can be used for obstacle avoidance or even basic object recognition.

Numerous approaches exist to estimate optical flow with a sequence of frames taken from a camera. The general technique computes the gradient (or laplacian) of the intensity of the image between consecutive frames. [5] gives a review of the existing techniques for computing optical flow. Another good review can be found in [117].

In [13] a quantitative performance analysis is performed using the nine most well-known optical flow computation algorithms. The outcome is that the Lucas and Kanade [68] and the Local Phase Based method [31] are the most reliable.

The main problem with optical flow is that it is computationally expensive since it usually requires a step for detecting relevant features to track between successive images (see for instance the Shi-Tomasi algorithm [99]). Also, another step is required to extract the ego motion of the robot from optical flow.

An interesting alternative could be the image interpolation technique described in [105]. This method does not compute the optical flow at each point of the image but instead estimates the motion of the whole image of two successive frames. Thus it is

directly suited to computing the ego motion of the robot, the drawback being that it discards the rest of the information contained in the optical flow.

1.4 Central Pattern Generators

The problem of generating trajectories for each joint of the robot resulting in a stable walking gait is non trivial. One simple way would be to directly send sine wave commands to each joint with carefully chosen phase shift between them for proper coordination. However this kind of simple control is difficult to modulate in a smooth way. Changing frequency, amplitude or phase shift online and smoothly becomes a challenge. Furthermore introducing sensory feedback while keeping the output trajectories smooth is not possible in the general case. One alternative is to use central pattern generators (CPG) for trajectory generation. CPGs are networks of coupled oscillators (dynamical systems) which can generate complex synchronized commands using simple input control signals. Amplitude, frequency and phase difference between oscillators can be modulated online with smooth transitions. They are very robust to perturbations, both in amplitude and phase shifts, and the generated trajectories smoothly converge back to a limit-cycle behavior. Thus including sensory feedback while generating smooth trajectories is implicitly handled by CPGs. For these reason we chose to use CPGs for low-level control in this thesis.

CPG models are increasingly used for different kinds of robots and types of locomotion such as insect-like hexapods and octopods ([50]), quadrupeds ([56]), swimming robots ([49]), and humanoids ([108]). For a more complete review of CPGs and their application to robotics control see [48]. The main benefits of CPGs for locomotion are their robustness against perturbations, the ability to smoothly modulate the shape of the oscillations with simple control signals and the possibility to integrate sensory feedback.

Integrating sensory feedback in CPG is a critical matter which is attracting a lot of attention recently. A precursor work in this field on the Tekken2 quadruped robot is reviewed in [24]. The authors implement a set of reflexes inspired from biology to maintain the balance of the robot walking on natural terrain. In [93], sensory feedback from touch sensors located on the feet of a quadruped robot is introduced to implement phase resetting of the CPG, and increase the stability of the robot on slopes and steps. The work in [18] introduces an Adaptive Frequency Oscillator (AFO), which when forced by sensory input coming from the robot accelerometer is capable of tuning itself to the resonant frequency to control the robot very efficiently in terms of energy consumption. A more detailed review of using sensory feedback with CPGs is given in Chapter 6.

Although sensory feedback is starting to be used extensively in CPGs, vision seems to be highly ignored. One reason for this seems to be that CPGs are used for low-level

locomotion control, where other sensors are traditionally preferred over vision. To the best of our knowledge, the most relevant pieces of work using vision with CPGs are the work of H. Kimura and his team ([36]) and the work of M.A. Lewis and his group ([65]), both tackling the problem of stepping over or around obstacles (see Section 1.2). Most of the efforts of the past decades have been dedicated to using CPGs for rhythmic locomotion pattern generation. Yet, periodic movements do not suit discrete tasks like manipulation or reaching. In [28] a system was presented which embeds both rhythmic and discrete motion generation in the same CPG architecture. The output movements can thus be pure rhythmic, pure discrete, or a combination of both. This system used as basis for the work presented in Chapter 2.

1.5 Thesis Context and Overview

From the previous sections, it appears clear that vision, although used extensively in robotics, has not been sufficiently explored, especially in the field of locomotion. The following observations can be made:

- While the exteroceptive part of robotics vision has been exploited thoroughly, the exproprioceptive part has been mostly ignored.
- Optical flow, although highly important to human and animal locomotion, is widely under-used in robotics locomotion, compared to object segmentation and stereo-vision, with the exception of flying robots.
- Some of the unsolved problems of robot locomotion, like keeping balance on difficult terrains, have been shown to be achieved by humans using vision. However, vestibular or other proprioceptive sensors are usually preferred to vision to solve these tasks in robotics.

This thesis places itself in this context. While not completely away from the more classical problems of vision and robotics, it aims at showing that vision can bring more to robot locomotion than what it currently does. We also investigate the reverse problem, namely what robot motion can bring to vision.

Most of the work presented in this thesis is meant to be applicable to any legged robot with very limited modifications. Throughout this book we will present applications to humanoid robots, quadrupeds, salamander robots and real humans. Among these robots, some are passive compliant, i.e. they are equipped with springs in series with the motors. Control of compliant legged robots is a hot topic nowadays. While compliant robots present great advantages (storage of energy, robustness to shocks, safety of use etc.), they are also more difficult to model accurately and to control since this additional number of passive degrees of freedom makes the robot under-actuated. Although this is not the main topic of this thesis, I will present approaches suitable to controlling compliant robots.

Although this thesis is articulated around vision and its applications in robotics, the problem tackled here are somewhat more general. Indeed, when investigating for instance postural control and gait stabilization of legged robots using vision, we have to study balance control in general. Thus this thesis presents novel approaches to tackle general problems of legged locomotion, where vision constitutes an additional tool to solve them.

Precisely, this thesis can be viewed as an attempt to answer the following questions:

A *What type of visual information is relevant for which subproblem of locomotion ?*
Different types of visual information might be useful for a particular problem but not for another. For instance, for precise feet placement, the exact three dimensional position of the target might be mandatory, while for balance control, the vision flow might be better suited. Furthermore some problems might be solvable using different kinds of visual information. Obstacle avoidance for instance, can make use of the precise three dimensional position of the obstacle or consider the deflection of the vision flow when approaching an obstacle. Throughout this thesis we try to define the minimal piece of visual information necessary to solve the given task.

B *What needs to be modulated in the locomotion patterns to allow adaptation of the nominal gait to rough terrain ?*
This question implies defining the different parameters that need to be tuned during locomotion according to the characteristics of the terrain. This modulation may act on the open parameters of the CPG (amplitude, couplings etc.) which in term will change the basic characteristics of the gait such as heading, stride length, angle of attack, phase lag between legs etc.

C *How to interface the visual system with the locomotor system ?*
Depending on the type of visual information and the task to solve, the way in which we translate visual signals into motor commands can differ quite a lot. In the case where the precise position of a target to reach is known for instance, one can provide it directly to an inverse kinematics solver defining attractors of the CPG. On the other side, the optical flow is a too high dimensional measure to be explicitly used. One would have to help the robot to map this information to a modulation of the parameters of the CPG to enable specific behaviors.

D *How to fuse information from the visual system and the other sensors ?*
Locomotion uses vision but also information from the vestibular system, tactile information, muscle fatigue etc. Integrating information from different sensors should improve performance compared to using only one of them. For this thesis we tried to analyze how well the robot can perform using only one sensory cue (proprioception, vision) and then add additional sensory inputs to quantify the performance increase.

E *How can locomotion be modulated to enhance the visual capabilities of a robot ?*
It is desirable to coordinate locomotion and vision in a symbiotic relation, each enhancing the other. If one manages to use locomotion to help the robot to acquire

better visual information, it can in turn improve the quality of the visual sensory feedback provided to the locomotion modulation system. For instance, we investigate head stabilization methods, which would filter out the vision flow due to the normal motion of the robot to better detect disturbances.

F *How can vision help controlling a compliant system ?*
In some cases, compliant systems have a major drawback: the lack of accurate information on the kinematics structure of the system at a given time. This happens for instance when not only the joints of the robot are compliant but also its structure. This is also the case for joints that are passively compliant and do not include encoders, or for encoders which do not take into account the spring-like part of the joint. We want to investigate if vision is suitable to control such compliant robots.

To answer these questions, my thesis work was shared between two laboratories: the Bio-Robotics Laboratory at EPFL and the Computer and Robot Vision Laboratory at IST. The Bio-Robotics laboratory is specialized in the control of locomotion of legged robots biologically inspired models of the spinal cord of animals: central pattern generators. Member of this laboratory use dynamical systems to control the movements of humanoids, quadrupeds, anguilliform and modular robots, as well as exoskeletons. Vision is however seldom used in this lab so far. The Computer and Robot Vision Laboratory is specialized in computer vision. While some people in this lab are pure computer vision experts (object detection etc.), most members try to solve robotics problems like manipulation, grasping and head stabilization with vision. This laboratory does however no research on locomotion. Being integrated in these two laboratories was for me a great opportunity to break the gap between legged locomotion control and computer vision research.

During my thesis, I have been included in two European projects.
The first one, RobotCub was centered around a humanoid child robot, the iCub. The project gathered a visual control community, mainly interested in grasping, object tracking and cognition. This project gave me insights into how to use vision at a high level to control the path of a robot crawling in a complex environment, and reach specific targets with its hands.
The second project, AMARSi was more aimed at implementing rich motor primitives on compliant legged robots: a quadruped named Oncilla and a biped named Coman. Thus the consortium of this project was mostly composed of locomotion control experts. While no vision laboratory was included in the project per se, participating in this project showed me how I could apply vision to low-level locomotion control tasks, namely biped postural control and quadruped gait stabilization. In the following two paragraphs I will present these two projects in more details.

1.5.1 The RobotCub Project

The RobotCub Project [94] is a European project funded by the European Commission through Unit E5: "Cognitive Systems, Interaction & Robotics". Its main goal is to

study cognition through the implementation of a humanoid robot the size of a three and a half year old child: the iCub (see Figure 1.1).

The approach of the RobotCub [94] project to tackle the Humanoid problem is a bit different than what is standardly done in the humanoid robotics community. Rather than directly tackling the immensely hard problem to build a fully grown up robot with full human cognitive abilities, this European initiative aims at building a child robot, in a way simpler, which would then learn and evolve the way a real child does.

The RobotCub project is a fully open project in the sense that the software, middle-ware and hardware of the iCub are open source. The RobotCub project ended in January 2010, thus since the beginning of my thesis work, the hardware and middle-ware of the iCub were almost final. The iCub has 53 degrees of freedom. A good number of them are allocated to the upper torso, especially to the hands (18 in total) to allow manipulation of objects. The iCub is strong enough to crawl on all fours and sit to free the hands for manipulating objects.

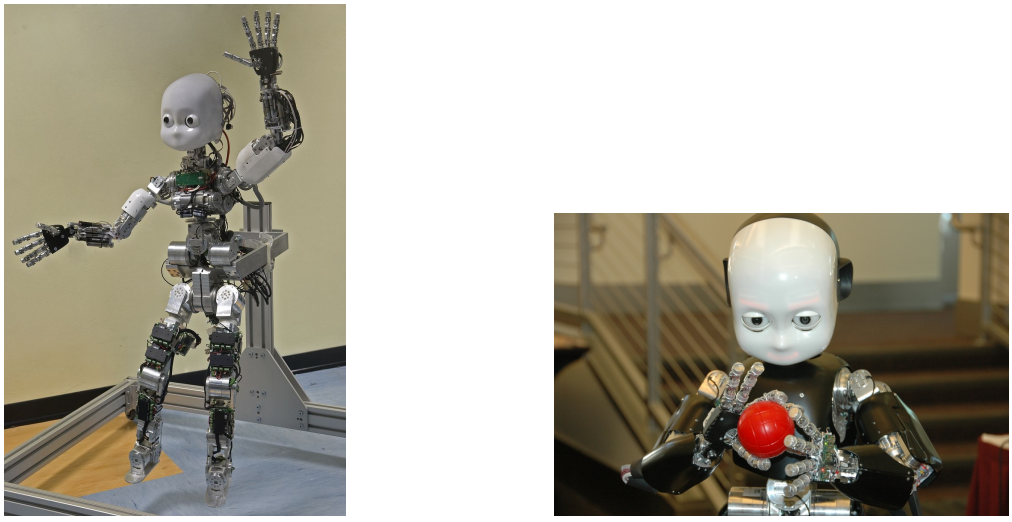


Figure 1.1: The iCub robot

The middle-ware of the iCub, YARP, was developed for the RobotCub project in an attempt to build a robust, durable, and reusable robotics platform. YARP philosophy is to allow clear decoupling between software modules and hardware devices in order facilitate the interfacing of the robot, but also the coordination of the different pieces of code developed by the different actors of the project. Communication in YARP is transport independent; details about the underlying network and protocol are hidden to the user.

This robotic platform is used for this thesis (see Chapter 2) and is available both at EPFL and IST.

1.5.2 The AMARSi project

The AMARSi project [9] is a European project funded by the European Commission through Unit E5: "Cognitive Systems, Interaction & Robotics". It aims at breaking the gap between the richness of motor skills of animals and robots. The final goal of the project is to design adaptive motor primitives and combine them to enable robots to perform rich movements. The AMARSi project is centered around dynamical systems, neural models and learning for the control part and compliant mechanics for the hardware part.

Two robots are being specifically designed for this project.
First a compliant humanoid robot, Coman, is being designed by the Italian Institute of Technology (IIT) who is part of the project. Coman implements passive compliant actuators and force sensors for active compliance in the legs and the arms of the robot. This platform was used for this thesis (see Chapter 5) and is available at EPFL.

Second, a cat robot, the Oncilla (see Figure 1.2), is being designed by the BioRobotics Laboratory at EPFL [16] specifically to enable fast adaptive running and locomotion on rough terrain. This robot implements three-segmented passive compliant legs with three actuated DOFs (hip protraction, hip abduction and knee), and a flexible toe element. It is relatively small ($\approx 25 \times 15 \times 18cm$), lightweight ($\approx 2 - 2.5kg$) and cheap ($\approx$ 15k€). The robot is powered by a 24V battery and is capable swing-stance frequencies of 3.5-4Hz at 90°. This platform was used for this thesis (see Chapter 6) and is available at EPFL.

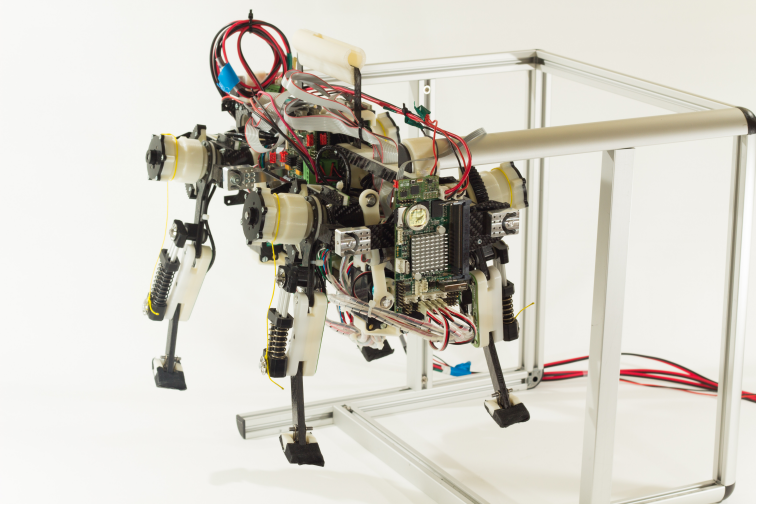


Figure 1.2: The Oncilla robot

1.5.3 General Control Framework and Structure of the Thesis

In this section we present the general control structure in which our work is included (1.3). While it is a very schematic description of the architecture is gives an idea of where each part of this thesis is located and their relations. It also shows how the work presented here can be merged into a single controller and used simultaneously.

Sensors are processed to extract meaningful information for the different tasks (layers 1 and 2). The high-level controller (3) uses processed sensory information to define general targets for the controller. These can be a path to follow through the environment or 3D positions of targets to reach for instance. The mid-level controller (4) translates these high-level goals into instantaneous control signals for the low-level controller. It generates modulations for the low-level controller, while ensuring certain constraints. Inverse kinematics for reaching or balance control algorithms are located in this layer. At the lowest level of control (5), we find central pattern generators, which are used throughout this thesis. The CPG dynamics directly defines the pattern of joint commands sent to the robot actuators (6). This architecture is modular, so that the lowest levels are usable without the upper ones. For instance, the CPG is fully capable of generating open-loop gaits in the absence of the mid and high-level controls. Note that layers 1 and 6 are physical layers (part of the robot hardware), while layers 2, 3, 4 and 5 are purely software layers. When these levels are active, they modulate the dynamics of the CPG to tackle a given task. The higher levels however do need the lower ones since their output need eventually to be translated into joint commands for the robot.

There exists direct links between the sensor processing level and mid to low-level controller, for head stabilization and balance control. Reverse signals from low to mid-level control exist, to enable phase dependent modulations of the CPG. This architecture being a closed loop, the motion of the robot commanded by the controller in turn influences the sensor values. The modified sensory feedback then influences the behavior of the controller. This leads to a complex dynamics of the whole control loop which is an important problem addressed in this thesis.

This thesis is organized in a top-down fashion, so that higher level controllers are presented first. This might seem surprising since as mentioned before, the upper levels need the lower levels while the opposite is not true. The main reason why I chose to present my work in this order, apart from the fact that it follows the chronological order of the work done during my thesis, is that the first chapters present results in rather classical problems of visual control, while the last ones tackle issues more rarely addressed with vision.

In chapter 2, I will present how to use vision at a high level on top of an existing low-level CPG controller. I will present how we used the embedded cameras of the iCub robot to scan the environment and plan a safe path towards predefined targets

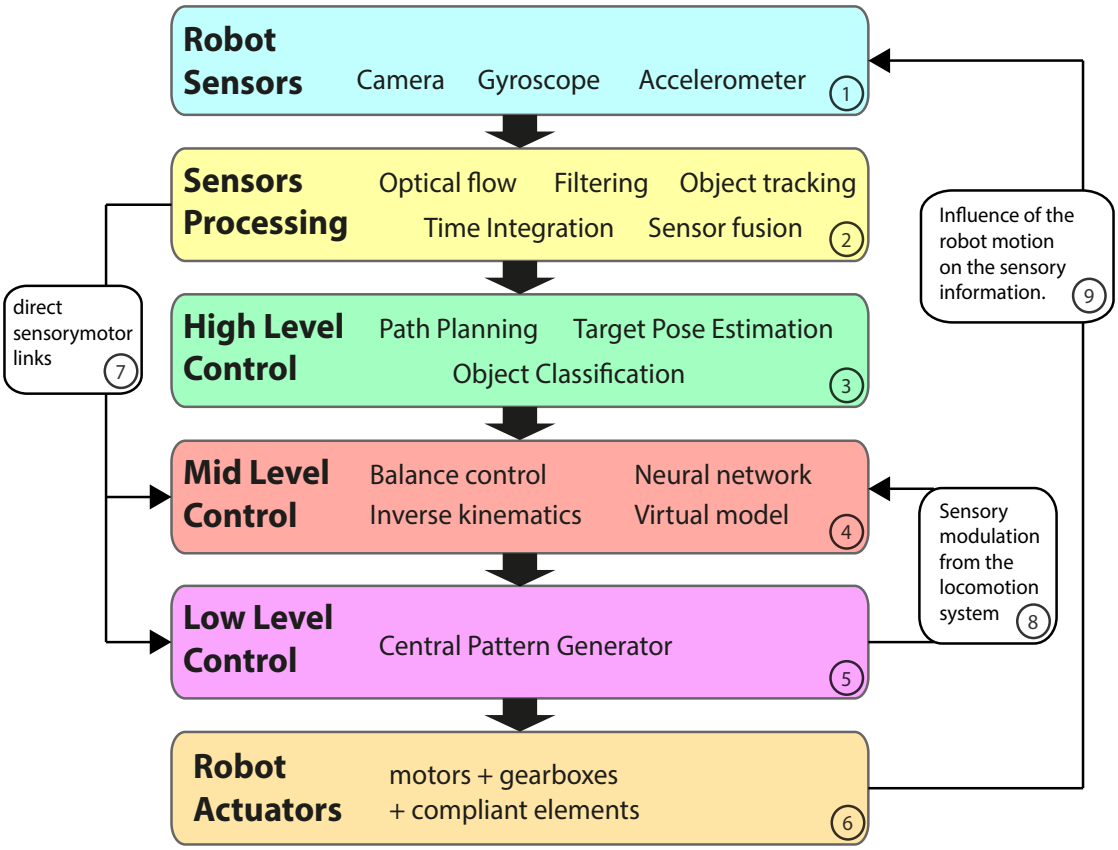


Figure 1.3: General structure of the control architecture for the work presented in this thesis

while avoiding obstacles. The path planning method is based on artificial potential fields which, given the estimated 3D position of obstacles and targets generate the instantaneous desired heading of the robot. When the robot reaches the target, inverse kinematics is used to actually reach it with its hand. An application of the same framework performing a discrete task, adaptive drumming, will also be demonstrated. The work I will present in that chapter is situated in sensor processing, high and mid layers of the architecture, while the lower level had already developed by a colleague (Sarah Degallier) at this time. The main contribution of this work is to present a fully integrated controller with vision used as main sensory feedback for a CPG based locomotion controller, to perform both rhythmic and discrete movements.

Chapter 3 investigates how robot motion can improve the quality of the visual information. It describes a novel approach to the head stabilization problem during periodic locomotion. The walking motion of legged animals causes displacements of the head which could cause a degradation of the visual information. To prevent that they perform head and eyes movements stabilizing their gaze. I will present a method to

stabilize the gaze of legged robots walking relying on vision as the sole sensory input. To compensate for the high latency of standard robot cameras, we designed a predictive algorithm based on adaptive dynamical systems to solve the gaze stabilization problem. This chapter is also located at a lower level than the previous ones since it uses direct links between the sensor processing layer and the oscillators controlling the head. However, the low-level CPG controlling the locomotion of the robot is unaffected by this method. The main contribution of this work is designing a system capable of performing gaze stabilization using no other sensors than vision, by exploiting predictive properties of adaptive dynamical systems.

Chapter 5 and 6 will bring the reader one level lower, by presenting modulations on the low-level locomotion controller by the sensors. Chapter 5 tackles the problem of discrete movements for postural control of a humanoid robot standing on a randomly moving platform, and Chapter 6 deals with stabilization of a periodic gait when a quadruped robot is walking on rough terrain. The goals of these two chapters are somewhat similar since they both aim at maintaining stable posture of the robot, whether it is standing or walking, when subject to perturbations from the environment (moving platform or uneven terrain). Thus it felt only natural to me to design a single control framework capable of handling both situations. This controller uses a neural network as direct mapping between the sensor values and modulations of the low-level control layer, and particle swarm optimization to learn stability. The main contribution of this work is a full framework for learning mappings between sensor values and CPG modulations in order to stabilize different robots performing both discrete (standing) and rhythmic (walking) tasks. This framework is also easily transferable to a real robotic platform.

Finally Chapter 7 will give a general discussion of the work achieved for this thesis, and guidelines for continuing this work in the future.

Chapter 2

Vision for High Level Control of Legged Robots

This chapter presents two applications of vision on top of a central pattern generator based locomotion controller for the iCub humanoid robot. First, a path planning algorithm is presented which uses vision to extract the position of goals and obstacles in the environment around the robot. Second an adaptive drumming controller is described, where the robot is required to hit on moving drums. These two applications shared the same low-level motion controller, which was developed by my colleague Sarah Degallier ([28], [26]). This controller is capable of generating both rhythmic and discrete motions, or even a combination of both, but was working completely in open-loop at the time of this work. Vision is used here to define high-level goals for this motion controller. Thus the work of this chapter is mainly included in the *Sensor Processing*, *High Level Control* and *Mid Level Control* layers (layers 2, 3 and 4 of Figure 1.3 in Chapter 1, Section 1.5.3).

The applications of vision presented in this chapter are also relatively classical in the robotics research literature. Vision constitutes the only sensor delivering accurate enough information about the surrounding environment to allow for efficient planning of collision-free trajectories for the whole robot or its end-effectors. Therefore it is used extensively for similar problems, and the added value of this chapter does not really lie in the novelty of the proposed algorithms. Instead the goal of this work is to present a complete architecture capable of using vision to achieve rich motion of a humanoid robot using a modular architecture with central pattern generators as base.

The work of this chapter aims at answering aspects of questions A, B, C and E, namely what are the relevant pieces of visual information for the task, how to interface vision with the locomotion system, what to modulate in the gait and how can the robot motion improve the visual information.

Most of the work presented here has been published in [37] and [26].

2.1 Introduction

Humanoids have inspired a lot of researchers and science fiction authors over the last few decades. Building a machine that would mimic humans with the same dexterity and robustness is a problem far from solved. The first step towards a fully functional humanoid robot is to enable it to move around its environment autonomously, identifying goals while avoiding collisions with obstacles.

This chapter presents our work on designing a closed-loop controller which, using only visual feedback from embedded cameras, allows a non-holonomic humanoid robot, the iCub, to locomote in a complex environment autonomously. It uses an infant like crawling gait, and reaches targets while avoiding obstacles using a potential field based planning. As a metaphor of a real infant, one can think of a child moving around a room towards toys scattered on the ground and grab them, while avoiding to bump into the furniture. We also apply our framework to the problem of adaptive drumming, where our child robot has to hit moving drums with a stick. This demonstrates the ability of our architecture to perform a combination of discrete and rhythmic movements with vision as sensory input.

2.1.1 Locomotion

The locomotion system we developed, already presented in [28] before, uses central pattern generators (CPG), i.e. networks of coupled oscillators inspired from the spinal cord of many animals. CPG models are increasingly used for different kinds of robots and types of locomotion such as insect like hexapods and octopods ([50]), quadrupeds ([56]), swimming ([49]), and humanoids ([108]). For a more complete review of CPGs and their application in robotics see [48]. The main benefits of CPGs for locomotion is their robustness against perturbations, the ability to smoothly modulate the shape of the oscillations with simple control signals and the possibility to easily integrate sensory feedback. Most of the efforts of the past decades have been dedicated to using CPGs for rhythmic locomotion pattern generation. Yet, periodic movements do not suit discrete tasks like manipulation or reaching. Our system embeds both rhythmic and discrete motion generation in the same CPG architecture.

2.1.2 Path planning

Numerous path planning techniques exist in the literature. Most of them use a geometric description of the environment and the robot. Grid based approaches overlay a grid on the map of the environment, reducing the path planning problem to a graph theory problem. Sampling techniques are currently considered the state of the art for a vast majority of motion planning problems. For a comparative description of grid based and sampling techniques, see [42]. Yet, both these methods require an exhaustive representation of the world to be efficient and a precise odometry estimation to be able

to achieve the computed road-map, which we do not have for our application.

Obstacle avoidance techniques are better suited to partially known environments. Examples of obstacles avoidance techniques include vector field histogram [17] which computes a subsets of motion directions and picks the best according to some heuristics and the dynamic window approach [34] which works in a similar way but in the velocity controls space.

An alternative method, at the border between path planning and obstacle avoidance techniques, is artificial potential fields [55]. The idea is to place artificial positive potentials on obstacles and negative potentials on the goal to attain, and navigate along the gradient of the potential field. The major problem of this method is its fragility to local minima, although some harmonic potential field functions have been developed to counter this weakness [96]. This method has not been developed specifically for non-holonomic robots, and some variants based notably on fluid dynamics theory [51] have been developed to cope with the constraints of these particular robots.

We chose to use an artificial potential fields approach for our application because it is (i) easily extensible to partial descriptions of the environment and dynamically changing environments, and (ii) it is computationally inexpensive, a necessary condition for online path planning.

Our approach does not claim to design a new state of the art motion planning algorithm. Instead, the goal of this work is to study the challenges that emerge when dealing with real legged non-holonomic robots. From this perspective we have developed a framework which integrates a vision tracking system exploiting the embedded cameras of the robot, a high-level motion planner based on artificial potential fields and acting on the low-level CPG controller, and an inverse kinematics solver for reaching. To our best knowledge approaches integrating all these features on a humanoid robot are very seldom in the literature. Examples include the work in [22] on the ASIMO robot which dealt with dynamical environments but where no vision was involved and a exhaustive representation of the world was provided to the robot. Other examples on different kinds of robots are found in [61] where a potential field approach was explored to plan the movement of a rover robot in an outdoor environment, and [109] which won the DARPA challenge consisting of having car robots locomote in a natural environment.

This study shows that online vision based navigation can be efficient even on a legged non-holonomic robot where vision and odometry estimation are strongly perturbed by the specificities of the quadruped gait (rolling effect etc.). It also shows an application of a fully autonomous high-level to low-level control system based on dynamical systems allowing rhythmic (crawling) and discrete (reaching) movements.

Finally one of the main concerns of this work is to match as closely as possible the constraints of the real robot. The gait that we designed implies a minimal radius of curvature of the robot when steering. The study presented here gives clues on how to adapt the parameters of the planning system to the actual constraints of the robot, when implementing on a real iCub. We also quantify the minimum radius of curvature

that a robot should have to achieve acceptable performance, which could be critical information when implementing new gaits for the iCub or even when designing the next generation of the robot.

2.2 Presentation of the architecture

Figure 2.1 shows the general architecture for crawling and drumming. For both case we use a layered architecture, with a low-level pattern generator, a middle level manager which role is to ensure the coherence of the commands sent to the generator, and a high-level planner embedding visual tracking, reaching and motion planning. This architecture is based on the work by my colleague Sarah Degallier and was the one used at the time of this work. Here the *Planner* is equivalent to the *High Level Control* layer in Figure 1.3 (Chapter 1, Section 1.5.3), the *Manager* corresponds to the *Mid Level Control* and the *Generator* to the *Low Level Control* layer. While Sarah Degallier was in charge of designing *Generator*, I developed modules of the *Planner* and *Manager*, which simply communicated with the *Generator*.

In this section, the locomotion architecture will be presented first, followed by a description of the vision and reaching systems and how they modulate the locomotion layer. In each paragraph, the specificities of the architecture and implementation for drumming and for crawling will be mentioned and we will finish with a description of the path planning algorithm developed for the first project.

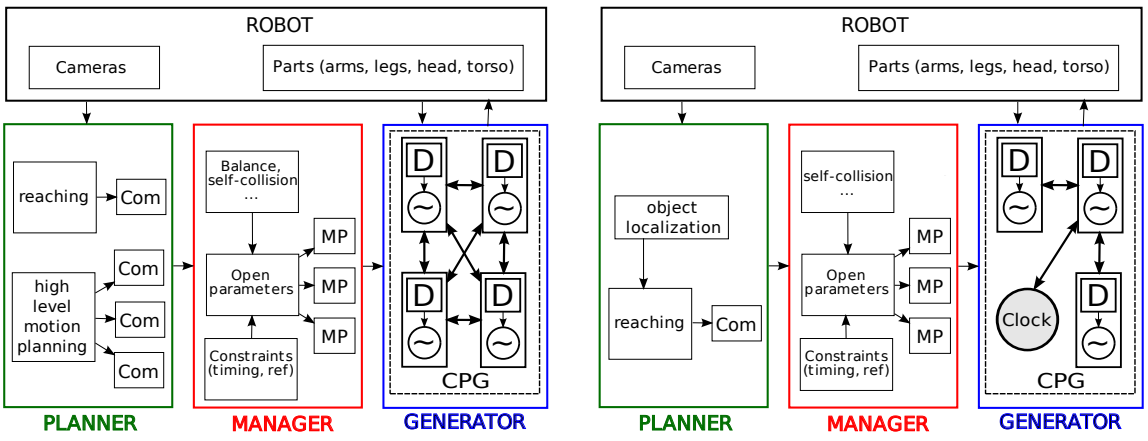


Figure 2.1: Main architecture for the motion planning of the crawling iCub (left) and for the iCub drumming (right)

2.3 Locomotion

The low-level locomotion controller presented here was developed by Sarah Degallier and not modified by the author of this thesis. It is built on the concept of central pattern generators (CPGs, see Section 1.4), that we take in the sense of a network of unit generators (UGs) of basic movements called motor primitives ([28]).

All trajectories (for each joint) are generated through a unique set of differential equations, which is designed to produce complex movements through the superimposition and sequencing of simpler motor primitives generated by rhythmic and discrete unit generators. The dynamics of the discrete movement is simply embedded into the rhythmic dynamics as an offset. These trajectories are sent as setpoints to the PID controllers of the motors. The discrete UG is modeled by the following system of equations:

$$\dot{h}_i = d(p - h_i) \tag{2.1}$$
$$\dot{y}_i = h_i^4 v_i \tag{2.2}$$
$$\dot{v}_i = p^4 \frac{-b^2}{4} (y_i - g_i) - b\, v_i \tag{2.3}$$

The system is critically damped so that the output y_i of Equations 2.2 and 2.3 converges asymptotically and monotonically to a goal g_i with a speed of convergence controlled by b, whereas the speed v_i converges to zero. p and d are chosen so to ensure a bell-shaped velocity profile; h_i converges to p and is reset to zero at the end of each movement.

The rhythmic UG is modeled as Hopf oscillator with the output of the discrete system as offset:

$$\dot{x}_i = a\left(m_i - r_i^2\right)(x_i - y_i) - \omega_i z_i \tag{2.4}$$
$$\dot{z}_i = a\left(m_i - r_i^2\right) z_i + \omega_i (x_i - y_i) + \sum k_{ij} z_j \tag{2.5}$$
$$\omega_i = \frac{\omega_{down}}{\mathrm{e}^{-f z_i} + 1} + \frac{\omega_{up}}{\mathrm{e}^{f z_i} + 1} \tag{2.6}$$

where $r_i = \sqrt{(x_i - y_i)^2 + z_i^2}$. When $m_i > 0$, Equations 2.4 and 2.5 describe an Hopf oscillator whose solution x_i is a periodic signal of amplitude $\sqrt{m_i}$ and frequency ω_i with an offset given by g_i. A Hopf bifurcation occurs when $m_i < 0$ leading to a system with a globally attractive fixed point at (g_i,0). The term $\sum k_{ij} z_j$ controls the couplings with the other rhythmic UGs j; the k_{ij}'s denote the gain of the coupling between the rhythmic UGs i and j and are set here to generate a trot gait. The expression used for ω_i allows for an independent control of the speed of the ascending and descending phases of the periodic signal, which is useful for instance for adjusting the swing and stance duration in crawling ([93]).

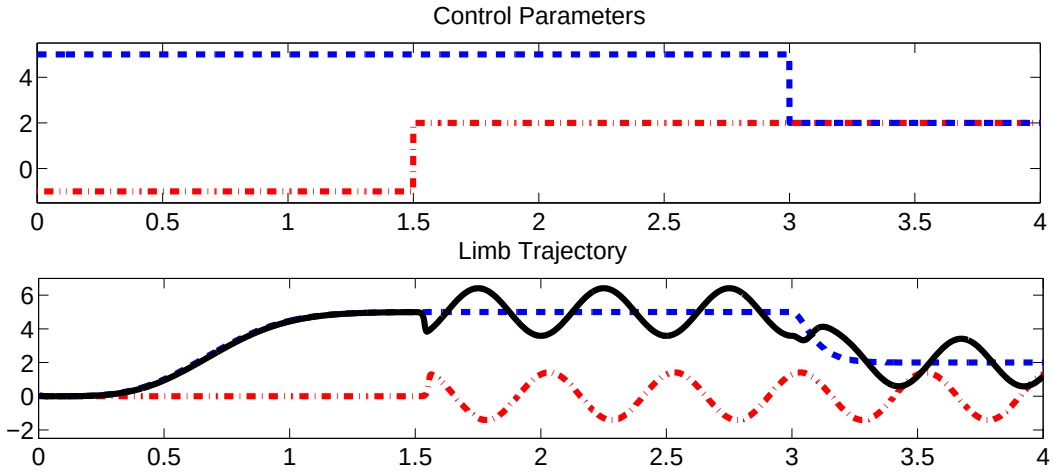


Figure 2.2: Unit pattern generators. Upper panel: control commands for discrete and rhythmic movements, that is the target position (in blue) and the amplitudes (in red), the frequency being not shown on the figure. Bottom Panel: The resulting discrete and rhythmic movements (resp. in blue and in red) and the trajectory embedding the two dynamics (black).

Qualitatively, by simply modifying on the fly the parameters g_i and m_i, the system can switch between purely discrete movements ($m_i < 0, g_i \neq \text{cst}$), purely rhythmic movements ($m_i > 0, g_i = \text{cst}$), and combinations of both ($m_i > 0, g_i \neq \text{cst}$) as illustrated on Figure 2.2. Different values for the k_{ij}'s lead to different phase relationship between the limb, i.e. different gaits for instance.

The low-level controller described here is used in a very similar way for crawling and drumming, the main difference being that for drumming, the unit generators are coupled to a clock defining the main rhythm of the motion. This clock, together with the setpoints of the oscillations, define the timing at which the stick will hit the drums. Note however that if the drum is moved during the course of the motion, the trajectory of the arm is updated to hit the new position, but the precise timing is not necessarily kept.

For more information on the low-level architecture, please refer to [28], [93], and [26].

2.4 Vision

For a robot to be able to navigate in an environment, it needs to be able to perceive it ; in our case see it. Similarly to be able to drum using drums that can be randomly positioned and even move, the robot needs to be able to recover the position of the

drums using its vision in a similar way a real instrumentalist would. The iCub is equipped with two cameras with the same two degrees of freedom as the human eye. As visual processing is not our main topic here, we chose to use a very simple marker based tracker, based on the ARToolKit Plus library [114]. Another reason for us to use this tracker is the fact that it does not use stereo-vision to compute the three dimensional position of a fixed sized marker which allows for faster tracking, an especially important feature when both the eyes and the head are moving during scanning. The obstacles and the goals are marked with different markers. The tracker is able to output the 3D position in the camera reference frame and the ID of multiple markers. On the real iCub robot, the tracker is able to detect an 8cm marker and its ID about 1.5m away. It is also very robust to changes of lightning. The position of the marker is translated to the robot root reference frame (attached to the waist) using forward kinematics.

For the first project, motion planning of the iCub crawling, the environment that we are considering here is corridor-like and composed of goals and obstacles (See Figure 2.3). We placed different markers on the goals and on the obstacles. We chose a corridor like environment so that most of the obstacles and goals appear in the field of view of the robot during locomotion. To compare the performance of our planning algorithm with different parameters, we also wanted to have a narrow environment in order to prevent the robot from turning back and have a finite dimension to have an upper bound of the performance.

For the second project, the iCub drumming, we placed the markers on the sides of the drums. This way the robot does not hide the marker when it drums. Since we are using impact feedback from the drums we also wanted to avoid having the markers directly on the drums since it would add noise to the feedback signal. The actual position of the center of the drum is obtained by defining an offset in the reference frame of the marker, and projecting it in the root reference frame of the robot using the 3D transformation matrix of the marker output by ARToolKit.

2.5 Reaching

Once the robot has detected a goal using the vision tracker described before, it has to reach it with its hand. While approaching the goal, the robot follows it with its head and eyes to keep it in the center of its vision field. This will allow him to make sure it does not loose the goal and to have a better precision on its position. The goal position is estimated using the vision tracker described in the previous section. Once the robot reaches a specific distance to the goal, it is considered “potentially reachable”. Starting from this point we use inverse kinematics to compute the joints angle of the 7-DOFs arm to achieve the target position, that is the position of the goal.

An inverse kinematics cartesian solver, (iKin) was designed specifically for the YARP

framework. This solver is based on the IPOPT (Interior Point OPTimizer) library [113], a library for large scale non-linear optimization. For our problem, given a desired end-effector position x_d in $\mathbb{R}^3$, the solver finds the joint configuration q in the 7 dimensional joint space $Q \in \mathbb{R}^7$ that achieves the nearest position $K_x(\boldsymbol{q})$ of the end-effector (here the hand of the robot):

$$\begin{aligned} \boldsymbol{q} &= \underset{\boldsymbol{q}\in\mathbb{R}^7}{\operatorname{argmin}}(||x_d - K_x(\boldsymbol{q})||^2) \\ \text{s.t. } & \boldsymbol{q_L} < \boldsymbol{q} < \boldsymbol{q_U} \end{aligned} \tag{2.7}$$

where q_L and q_U are the lower and upper joint limits of the arm of the robot. For more details about IPOPT and the non-linear solver see [113].

It is then possible to compute the euclidean distance between the desired and achieved positions $\rho(x, x_d)$ and set a threshold ϵ defining the reachability of the goal. Once the goal is "reachable", the robot moves its hand to the computed position q that achieves x using the discrete system described in Section 2.3.

A similar approach has been taken for drumming. A modified version of the inverse kinematics solver has been developed to include the drumming stick in the kinematic chain when solving the reaching position. The robot detects the markers and computes the joint positions of its arms to reach the center of the drum with the tip of the drumming stick. These positions then defines the attractors of the oscillators so that the tip of the stick is at the center of the drum at the time of impact.

2.6 Planning

The purpose of the planning module is to have the robot navigate in a world composed of multiple goals and multiple obstacles. This part is obviously specific to the first project, crawling and is not present for drumming. The input of this module is a set of 3D positions of goals and obstacles sent by the vision tracker described in Section 2.4 and expressed in robot coordinates. The field of view of the cameras of the iCub is relatively small ($\alpha \approx 45°$) which gives a very small amount of information to the robot about its surroundings. To counter this limitation, we make the robot scan the environment by rotating its head and eyes from left to right. An egocentric partial map of the environment is built by merging the areas scanned over a full rotation of the head. The head oscillations are coupled with the limbs movements to have the scanning speed depend on the locomotion speed. The frequency of the head oscillations was set to half that of the limbs (one head rotation every two steps). This scanning made it possible to extend the vision field of the robot to $\theta \approx 120°$ (see Figure 2.4).

Every time a head scanning is finished, a partial map of the environment is generated and attractive potentials $U_a(\boldsymbol{p})$ are placed on the goals and repulsive ones $U_r(\boldsymbol{p})$ on the obstacles, $\boldsymbol{p}$ being the robot 2D position on the map (note that since the map is an egocentric one, $\boldsymbol{p} = (0, 0)$). Figure 2.3 shows an example of a partial map of the

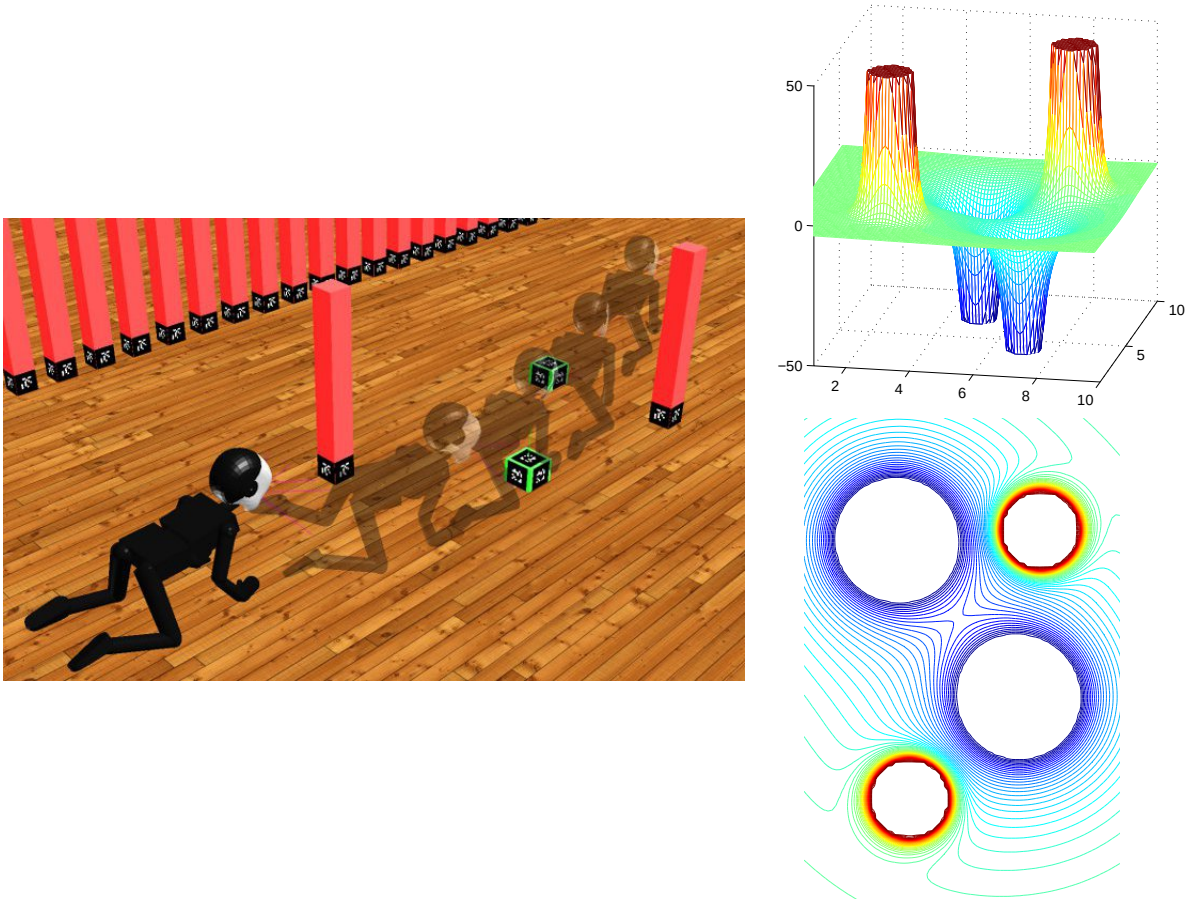


Figure 2.3: Snapshot of a Webots world with the iCub at $t = t_0, t_1, t_2, t_3, t_4$. The associated potential built by the robot at $t = t_0$ (in a theoretical ideal case) is represented on the right (the scales are different). The field of action of an attractive potential is wider than that of a repulsive one due to the maximum distance of action $\rho_0 = 2$ and $k_r = 4$ of the repulsive potentials (see Equation 2.8)

environment and the potential field associated to it.

Usual potential fields methods have a unique goal and thus define the attractive potential and the corresponding attracting force proportional to the distance to the goal or even to a power of it. Having multiple goals, we cannot define it this way since the robot would keep oscillating between goals without ever reaching one. Instead, we chose to define the attractive and repulsive forces ($\vec{F_a}$) and ($\vec{F_r}$) created respectively by the goal and obstacle potentials as:

$$\begin{aligned} \vec{F_a} &= -\nabla U_a(\boldsymbol{p}) = -\xi \frac{1}{\rho(\boldsymbol{p})^{k_a}} \vec{u} \\ \vec{F_r} &= -\nabla U_r(\boldsymbol{p}) = \begin{cases} \eta \frac{1}{\rho(\boldsymbol{p})^{k_r}} (\frac{1}{\rho(\boldsymbol{p})} - \frac{1}{\rho_0}) \vec{u} & \text{if } \rho \leq \rho_0, \\ 0 & \text{if } \rho > \rho_0. \end{cases} \end{aligned} \tag{2.8}$$

where :

- $\rho(\boldsymbol{p})$ is the euclidean distance between the origin of the potential and the robot.
- ρ_0 is the maximum distance of influence of a repulsive potential.
- k_a and k_r are positive factors that determine the curvature of the potential surface.
- ξ and η are positive scaling factors.
- $\vec{u} = \nabla \rho(\boldsymbol{p})$ is a unit vector oriented away from the origin of the potential and towards the robot.

The resulting force $\vec{F_\Sigma}$ that applies on the robot is then simply:

$$\vec{F_\Sigma} = \sum_{i=0}^{n} \vec{F}_{a_i} + \sum_{j=0}^{m} \vec{F}_{r_j} \tag{2.9}$$

n being the number of goals and m the number of obstacles.

The robot moves then of a small distance following $\vec{F_\Sigma}$. Its displacement $\vec{D}$ and angle of rotation ϕ can be defined as :

$$\begin{aligned} \vec{D} &= \Delta \frac{\vec{F_\Sigma}}{||\vec{F_\Sigma}||} \\ \phi &= atan2(\vec{r}^{\perp}.\vec{u}, \vec{r}.\vec{u}) \end{aligned} \tag{2.10}$$

Where where $\vec{r}$ is the current direction of motion of the robot and Δ is a small distance to be defined and $^{\perp}.$ is the perp-dot product.

Here we only compute ϕ explicitly and let Δ be the distance achieved by the robot between two refreshing of the potential field (between two full scans). In theory, the

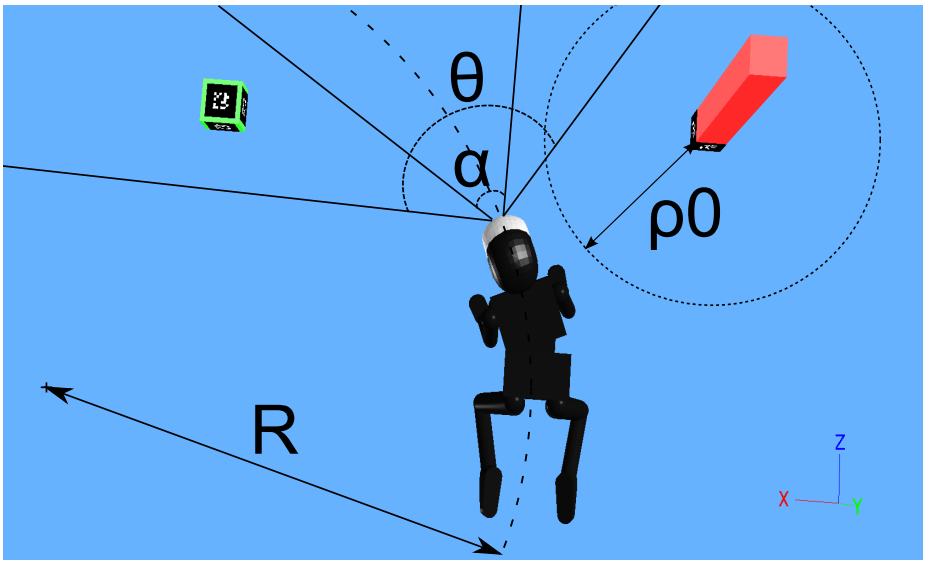


Figure 2.4: A snapshot of the Webots world with annotations of the important quantities. The red pylon is an obstacle, the green cube a goal (notice the ARToolKit markers on them). α is the field of view of the robot, θ the extended field of view due to the scanning process, and R_{min} the minimum radius of curvature

actual rotation angle of the robot corresponds to the torso roll angle of the robot (see Figure 2.4), but may be somewhat different on the real robot due to sliding of the limbs on the ground.

The values of k_r, k_a and ρ_0 in Equation 2.8 influence strongly the shape of the potential field. Figure 2.5 shows this influence for two different values of ρ_0. A potential with a low k (k_r or k_a) has a slighter slope, and thus a larger range of influence than one with a big k. By varying ρ_0, one can explicitly limit the range of influence of an obstacle potential. Setting a low ρ_0 is particularly useful if one wants the robot to be able to squeeze in between obstacles. Setting a high k_r has a similar effect, while also changing the slope of the potential field. Section 2.7 presents a study of the influence of these various parameters on the performance of robots with different minimum curvature radius.

2.7 Results

2.7.1 Results for crawling:

The main questions we address here are (i) How well does our planning system perform for robots with different minimum radius of curvature R_{min}, (ii) how do the different parameters of the potential field equations described in Section 2.6 influence the

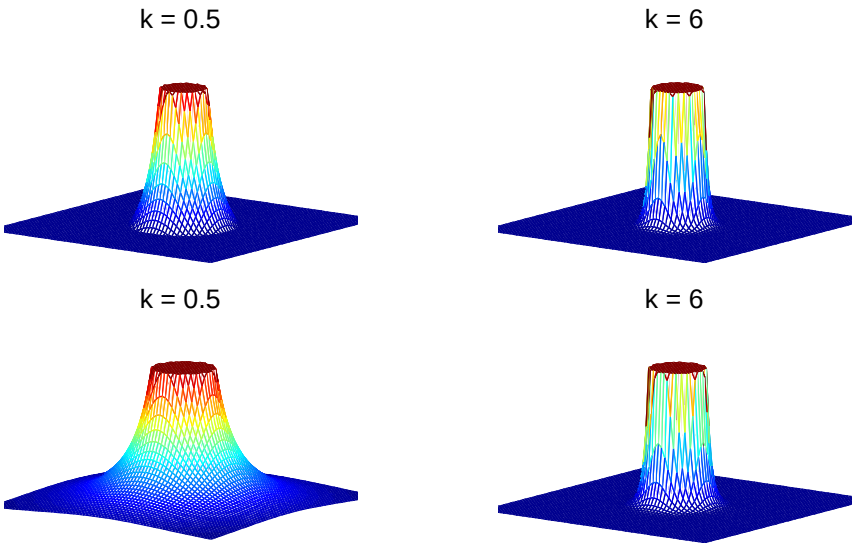


Figure 2.5: Influence of k (k_r or k_a) on the shape of the potential field for $\rho_0 = 2$ (top) and $\rho_0 = 10$ (bottom). The slope of the potential surface increases with k, while ρ_0 allows an explicit limitation of the range of influence of the potential

performance and how are they related to the values of R_{min}, and (iii) what minimum value of R_{min} should we achieve in order to reach good performance. This last point is particularly important when designing a turning gait for a robot since it helps finding a compromise between the performance of the locomotion and that of the planning. For instance, if one cannot find a stable turning gait leading to a small minimum curvature, one could decide to have the robot turn on itself by performing a series of maneuvers, at the expense of the speed of locomotion.

The performance of the system was measured by the number of reached goals versus the number of collided obstacles. In order to study the influence of the various parameters on the performance of the motion planning algorithm with the constraints of the real robot we used a two stage simulation approach: first using a 2D simulator, having enough simplicity and speed to test a wide range of parameters and then using the physics based robotics simulator Webots ([116]). Implementation of the crawling locomotion system and the visual based reaching has been done and will be discussed at the end of this section. The video at [3] illustrates the work of this chapter.

2D simulations

We performed a series of systematic tests using a simple 2D simulator on the following parameters: k_a, k_n, ρ_0 and R_{min} (the minimum radius of curvature of the robot). In this simulator, no vision is involved but the field of view of the robot is constrained

geometrically. Thus obstacles and goals are only "seen" by the planning algorithm if they are in an area corresponding to a field of view of 120°, with a depth of two meters, from the robot position and along its orientation. These values are coherent with the real robot properties. We generated 70 corridor-like worlds of dimension 4×40 m, containing 10 goals and 15 obstacles each, and enclosed by walls of obstacles. The goals and obstacles were randomly positioned with the only condition that the distance between each of them was at least 1m. This is to ensure a rather uniform distribution of goals and obstacles and avoid worlds with conglomerates that would be impossible for any parameters and thus would lead to similar scores for all trials. The parameters were taken in the following sets: $k_r = \{0.5, 1, 2, 4, 6\}$, $k_a = \{0.5, 1, 2, 4, 6\}$, $\rho_0 = \{1, 1.3, 1.5, 2\}$, $R_{min} = \{0.7, 1, 2, 3, 4, 6\}$ (42000 runs). The results of these systematic tests are presented in Figure 2.6.

The top left graph shows the mean number of reached goals and collided obstacles over all runs for each value of R_{min}. As can be expected the smaller the minimum radius of curvature the better the performance. The fact that the number of obstacles collided is lower for $R_{min} = 6$ than for $R_{min} = 4$ is due to the corridor shape of the world tested. Indeed, for $R_{min} = 6$ the robot moves almost in straight line and thus the probability to collide with the walls is reduced. Interestingly for $R_{min} < 1$ the performance does not increase so much anymore, suggesting that a minimum radius of curvature of 1 should be sufficient to achieve near-optimal performance.

The top right surface plot shows the influence of ρ_0 on the number of goals reached and obstacles collided. For a small R_{min}, the value of ρ_0 has barely any influence on the performance. The small radius of curvature of the robot allows it to avoid obstacles even if they influence its motion only very late (ρ_0 small). For higher R_{min} however, the performance strongly decreases with ρ_0. This time the robot can only avoid obstacles if it can anticipate enough (ρ_0 big).

The bottom two graphs show the influence of k_r and k_a on the performance for a small and a big value of R_{min}, and for $\rho_0 = 2$. When the radius of curvature is sufficiently small, the values of k_r and k_a are, like ρ_0 in the previous graph, not critical. This independence of the parameters for small R_{min} is a good feature of the planning algorithm for real robotics applications, since it means that the system does not significantly depend on specific parameters choices. Very small values of k_r and k_a lead to slightly lower performance, since the robot cannot get near enough obstacles to perform quick maneuvers, which would be made possible and safe by its small radius of curvature.

For big R_{min} the number of collided obstacles mostly increases with the value of k_r, since for big k_r the influence of the obstacle potentials decreases rapidly with the distance and so the robot cannot anticipate enough to cope with its big radius of curvature. A less intuitive observation is that the number of reached goals decreases for small values of k_a. This can be explained by the fact that, where several goals are

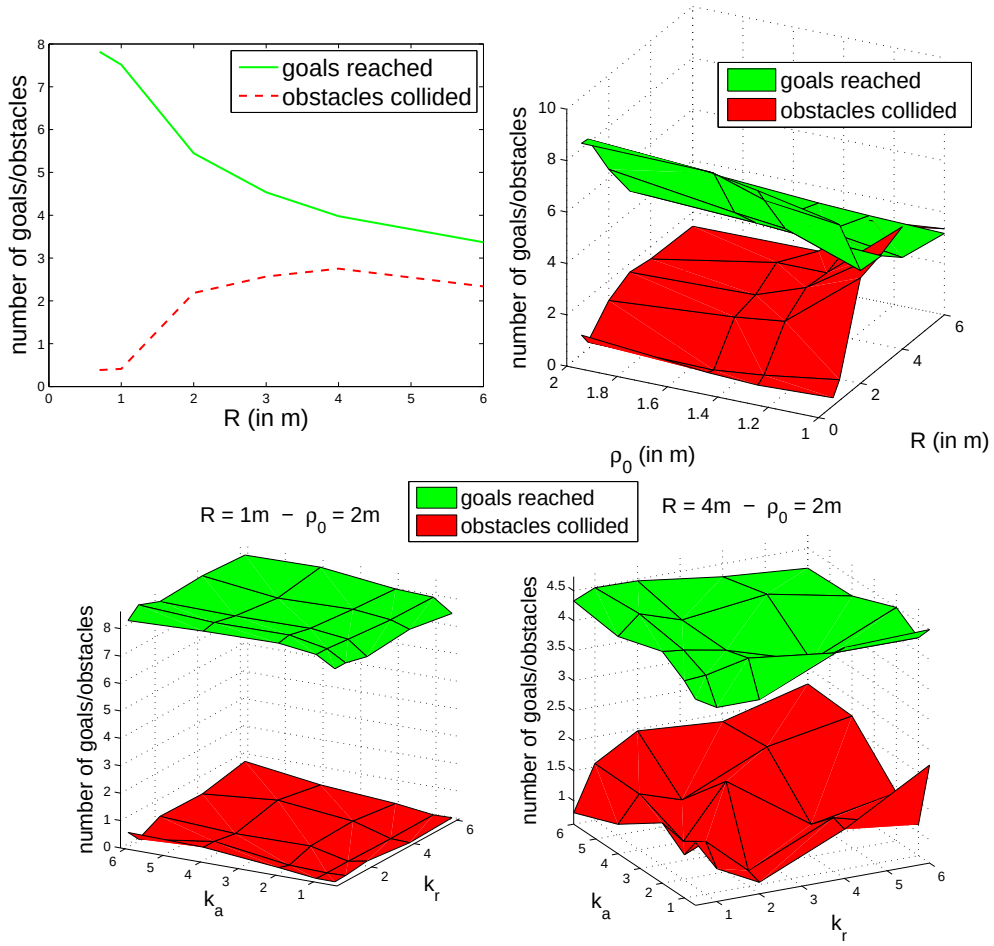


Figure 2.6: Results of the systematic tests using the 2D simulator. Top left: number of reached goals and obstacles collided over the whole pool of tests. Top right: influence of ρ_0 on the performance. Bottom left: influence of k_a and k_r on the performance for a small radius of curvature ($R_{min} = 1$) for $\rho_0 = 1$. Bottom right: influence of k_a and k_r on the performance for a big radius of curvature ($R_{min} = 4$) for $\rho_0 = 2$

.

in the field of view of the robot, and k_a is small, their influence would mostly balance until one is significantly nearer than the other. At that point however, with a big R_{min}, the robot would not be able to turn fast enough to reach the nearest one. This happens in Figure 2.7 for $R_{min} = 4$ for the 2D simulator. At $y \approx 12$ the robot passes in between two goals without reaching any of them.

Webots simulations

The observations made in the 2D simulator are useful to adapt a potential field based planning algorithm to the constraints of a real non-holonomic robot. But first we have to check that the behavior of a real robot would match that of the 2D simulation, at least concerning curvature radius issues. In the physics based simulator Webots, detection of the obstacles and goals is not geometrical anymore as in the 2D simulator but uses the perspective projection Webots cameras, the "eyes" of the robot, to perform visual processing using the ARToolKit based marker tracker described in Section 2.4. Hence detection is not deterministic anymore but subject to noise in the position extraction of the markers. Locomotion of course is significantly different since it uses the CPG based system described in Section 2.3 and not a simple translation like in the 2D simulator. This also induces noise in the vision tracking due to movements of the head and a high variance in the potential field generation since markers are constantly entering and escaping the field of view of the robot, causing modifications in the potential field. To cope with these issues, we performed noise filtering at the vision level and introduced a short term memory at the planning level. This memory introduces damping in the changes of the potential field and thus prevents the robot from constantly changing direction.

Due to the complexity of the simulator + locomotion + vision tracker + planning system, we only performed a limited amount of tests, to prove the efficiency of the whole framework and show that the results match that of the 2D simulation. We chose a world that gave significantly different results for different values of R_{min} in the 2D simulations. We run 5 runs for each values of $R_{min} \approx 1, 2, 3, 4$. We could not find a stable gait leading to $R_{min} < 1$ (the robot would not move) or $R_{min} > 4$ (the robot would move in straight line). A significant difference between the way the radius of convergence is computed in the 2D simulator and in Webots is worth mentioning. In Webots, turning is achieved by changing the torso roll angle (see Figure 2.4) and modifying the amplitudes of the left and right limbs accordingly. However, the robot cannot reach its maximum turn angle at once since it would cause a lot of sliding and big constraints on the motors. Thus at each step the turn angle increases by a small amount, and so the radius of curvature is not constant, unlike in the 2D simulator. The values of R_{min} given before are the curvature after the maximum turn angle has been reached, which may be different from the actual turn angle while navigating.

Figure 2.7 (top two graphs) shows the performance of the planning for different

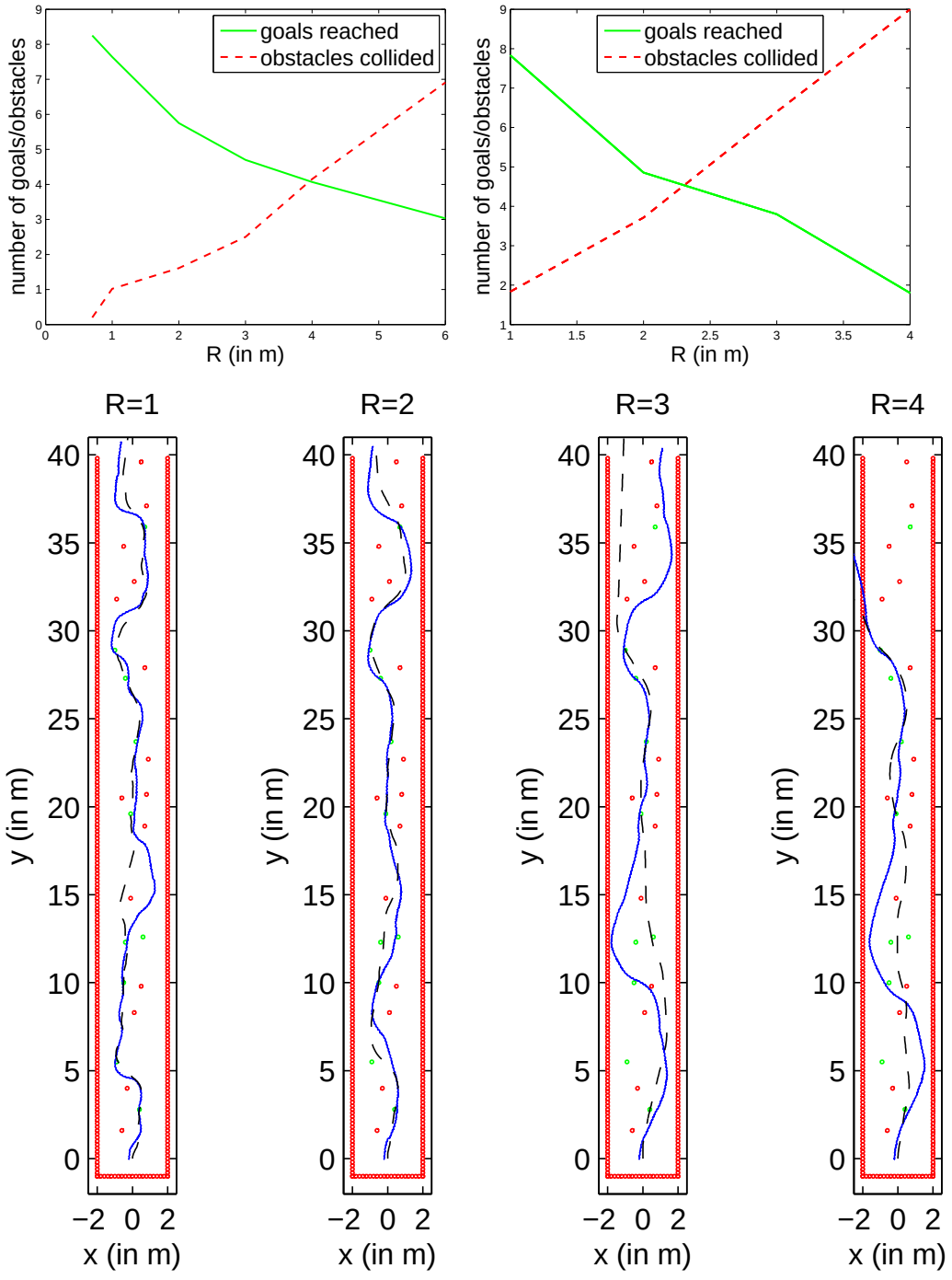


Figure 2.7: Comparison of the performance of the planning algorithm for different radius of curvature in one world using the the 2D simulator (top left) and Webots (top right). Comparison of trajectories with similar performance in Webots (blue solid line) and the 2D simulator (black dashed line) for different values of R_{min} (bottom figure).

values of R_{min} in Webots and in the 2D simulator. Interestingly the relation between the maximum curvature and the number of goals reached and obstacles collided is qualitatively the same as in the 2D simulator. Thus the 2D simulator is a good approximation of the Webots simulation which should be a good approximation of the behavior of the system on the real robot. However, quantitatively, results are different, the values in the 2D simulator corresponding approximatively to those in Webots for $2 \times R_{min}$. This is mostly due to the imperfect match between the curvature in both simulators, as discussed before.

Overall the planning algorithm proved to solve well the planning problem with the proper parameters. For $R_{min} = 1$ the robot was able to reach 9 goals out of 10 and collide with no obstacle (even reach 10 in the 2D simulator). The fast online refreshing of the potential field during locomotion allows the robot to handle dynamical environments. The video [3] shows the iCub navigating in a Webots world with only one goal moved around manually. The obstacles were also moved during this experiment. In the end the iCub was able follow the moving goal while avoiding the obstacles.

Implementation on the iCub

Finally we implemented the crawling and visual based reaching mechanisms on the real iCub robot. We did not yet implement steering and thus did not test the planning algorithm on the iCub. The experiment consisted in having the robot crawl for a couple of meters, then detect a marker placed on the ground, follow it with its head and reach it with its right arm. Crawling proved very stable even though controlled in open-loop, and the robot was able to switch instantly from rhythmic to discrete movements when reaching. Visual detection and tracking showed good performance and the robot seldom lost track of the marker before reaching it. The attached video presents this experiment. Figure 2.8 shows the output of the CPG and the actual trajectories of the four controlled joints of the right arm (the other limbs are qualitatively similar).

The robot followed very closely the commands sent by the CPG when crawling and reaching. The small offset between the encoders and the CPG output when reaching for joint 0 is due to the velocity limit of this joint. The small oscillation when crawling for joint 2 is due to a mechanical coupling of the three shoulder joints, but was not problematic for crawling. Figure 2.9 shows snapshots of the the robot crawling and reaching a marker placed on the ground.

2.7.2 Results for drumming

Adaptive drumming experiments were performed both in simulation and with the real robot. In both cases drums were moved around manually while the robot was performing the drumming task. While in the simulation experiment, vision was not

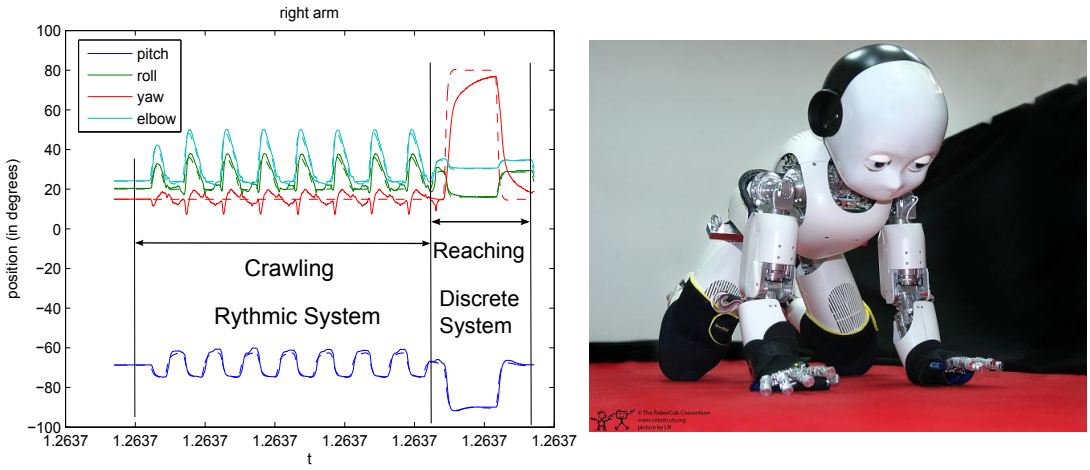


Figure 2.8: Left: output of the CPG (solid line) and encoders of the four controlled joints of the right leg and right arm during crawling then reaching. Right: picture of the iCub crawling

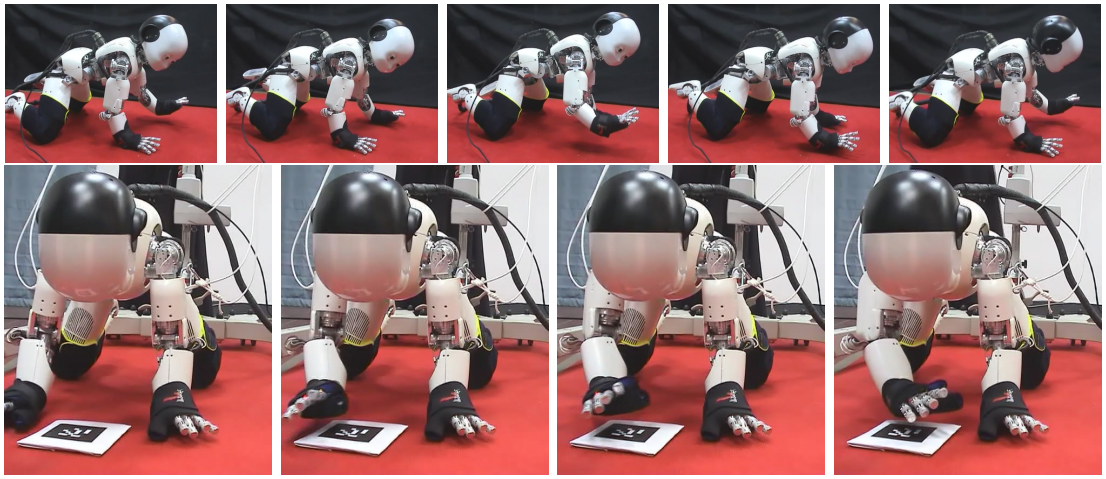


Figure 2.9: Snapshots of the robot crawling and reaching an ArtToolkit marker placed on the ground.

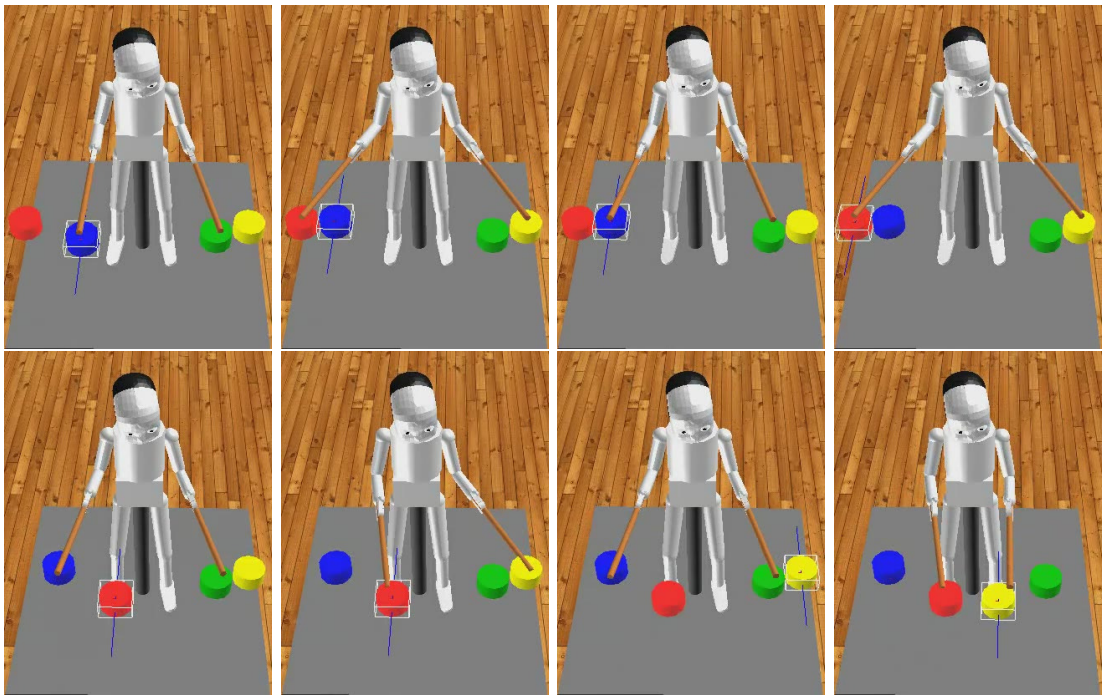


Figure 2.10: Snapshots of the simulated iCub drumming on moving drums. The position of the drums is extracted directly from the simulator without using vision, and used as input to the inverse kinematics module, which modifies the offsets of the CPG oscillations. The targets of the CPG are set such that the robot alternatively hits the blue and green drums and then the red and yellow ones.

used and the position of the drums was directly used as input to the reaching controller, in the real robot experiment, vision was exploited by placing ArtToolkit markers on the drums. In both cases the position of the drum is used as input to the inverse kinematics (reaching) module described in Section 2.5, with a modified kinematic chain to include the drumming stick. Figure 2.10 and 2.11 show snapshots of the the iCub drumming on moving drums in simulation and on the real robot.

On the real robot, the performance was not ideal due to the whole vision + inverse kinematics loop introducing big delays in the control. However, the robot was still able to successfully detect the moving drums and hit them with its drumming stick.

2.8 Conclusion

We have presented in this chapter a full system to allow a humanoid to navigate in a simplified environment using only its vision to get knowledge about its surroundings. The low-level locomotion mechanism, the *Generator*, uses coupled non-linear oscillators

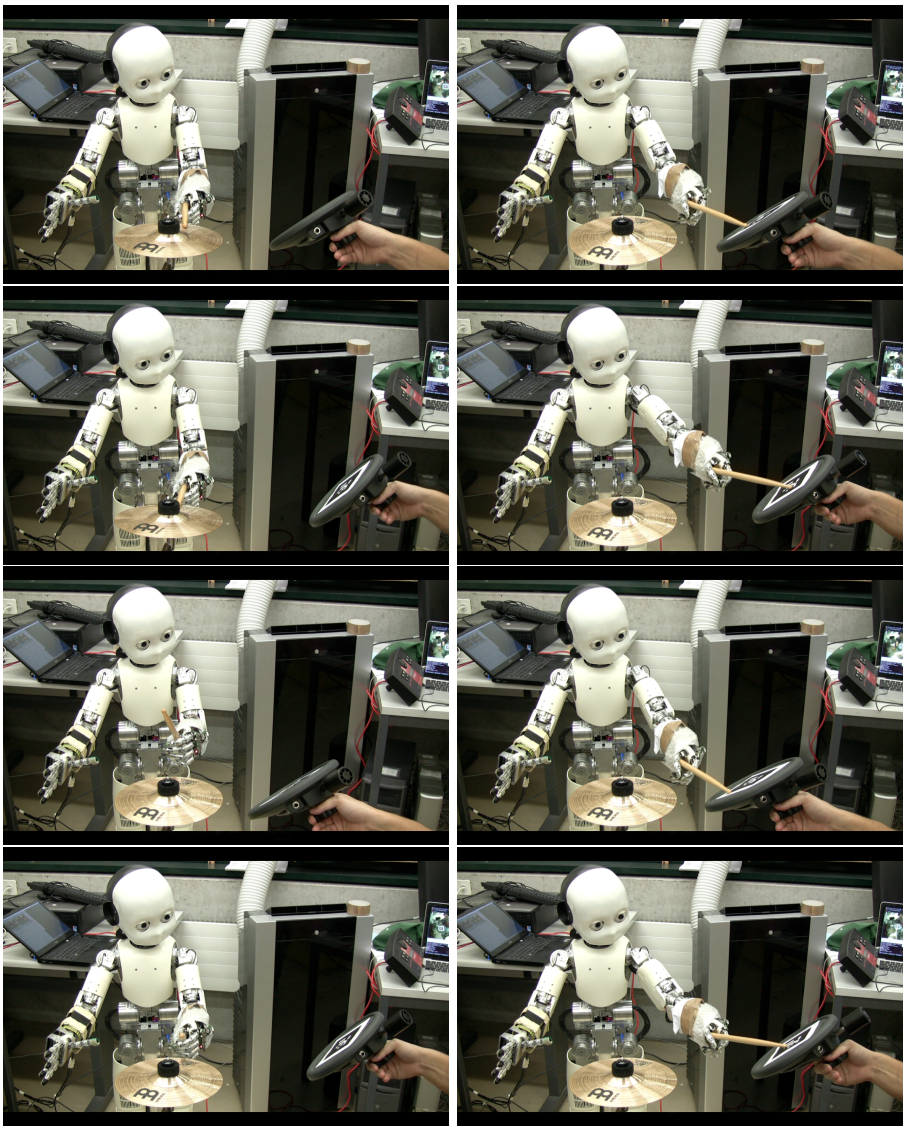


Figure 2.11: Snapshots of the real iCub drumming on a moving drum. The robot was asked to required to drum once on the fixed cymbal and once on the moving drum. The position of the drum is extracted using vision with an ArtToolkit marker placed on the drum, and used as input to the inverse kinematics module, which modifies the offsets of the CPG oscillations.

(CPG), to generate complex locomotion patterns using simple control inputs. These inputs are modulated by the *Manager* to cope with internal constraints. High level commands are sent by the *Planner* to the manager. This locomotion framework is able to perform rhythmic movements, for crawling, and discrete movements for reaching. At the highest level, we designed a motion planning system based on potential fields and using the visual cues provided by a marker based tracker. The whole locomotion + vision + motion planning + reaching system is thus fully autonomous. For this work, we kept the names used by Sarah Degallier to describe the layers of the architecture, since we built on top of her existing one. It should be noted that the *Generator* is equivalent to layer 5 presented in Figure 1.3, the *Manager* corresponds to layer 4 and the *Planner* to layer 3.

We successfully used a simple marker tracking algorithm using a single camera to locate goals and obstacles in the surrounding environment of the robot, thus answering part of question A described in Section 1.5 of the introduction of this thesis. We considered question C when interfacing the visual information to the path planning and reaching modules, subsequently interacting with the low-level controller. To enable turning the amplitudes of the hip oscillators as well as the offset of the trunk roll oscillator were modified, while for reaching the targets of the discrete part of each oscillator of the arm were changed by the inverse kinematic module. This answers question B for these particular tasks. Finally, when coupling the head movements to the oscillations of the legs, we addressed aspects of question E, since this allowed to widen the field of view of the robot.

We proved the efficiency of our system on a realistic robotics environment and using a 2D simulator to study the influence of the various parameters of the potential field equations on the performance while respecting the constraints of non-holonomic robots. We showed that for a small radius of curvature, the system is very stable to changes in parameters, while for big radius of curvature, setting the values of k_p and k_n low and ρ_0 high allows the robot to anticipate more and compensate for its big curvature.

One specificity of our work worth mentioning is the shape of the environments tested: corridor like. We suppose that the behavior of the system would be similar in different shape of environments but proving it is left for future work.

Implementation on the real iCub showed promising results, the robot being able to crawl, track a marker on the ground while crawling and finally reach it.

Further work should include more systematic tests of the planning system in the realistic robotics environment and on the real robot.

Chapter 3

Robot Motion for the Sake of Vision: Gaze Stabilization

This chapter presents an application of using vision directly coupled with an adaptive dynamical system to stabilize the gaze of robots performing periodic locomotion or tracking periodically moving objects. This an example of using the motion of the robot to enhance the quality of the visual feedback, since gaze stabilization decreases motion blur in the image. The work presented in this chapter is hence part of the *Robot Sensors*, *Sensor Processing* and *Low Level Control* layers (layers 1, 2 and 5 of Figure 1.3 in Chapter 1, Section 1.5.3). The oscillator designed for this work is here seen as part of the *Low Level Control* since it directly outputs commands for the joints of the robot. However the locomotion controller can run without the head stabilizing controller being active, while the opposite is not true. Our method for gaze stabilization requires either the robot or the observed environment to be subject to periodic motions and cannot for instance deal with discrete robot perturbations (pushes, singles steps etc.). Hence the gaze stabilization problem is viewed here as a slightly higher level problem as the ones described in the next two chapters dealing directly with the control of locomotion.

The problem of gaze stabilization is also by essence linked with vision. However vision is not always the main sensory information used to tackle this problem, vestibular sensors in the head being usually preferred for their low latency, and vision being usually used as an error measure. Our approach is novel in this sense since it uses exclusively vision as sensory input, and deals with the high latency of the visual sensors by exploiting the prediction abilities of the adaptive dynamical system designed.

We developed a motorized wireless camera to validate our approach, thus developing parts of the *Robot Sensors* layer (layer 1 in Figure Figure 1.3). This is the only piece of work in this thesis directly developing a physical layer.

This chapter is the main one addressing question E, i.e. how robot movements can be used to enhance the quality of the visual information. The work presented here aims at stabilizing the head of the robot when walking, hence increasing the quality of the camera images. It also addresses aspects of questions A and C, namely what part of the visual information to use for the task and how to interface vision with the motion controller.

The work presented in this chapter was published in [38] and [39].

3.1 Introduction

Vision is, for animals and robots, the most versatile sensor to provide information about the surrounding environment. However, vision is most efficient when the image (and thus the gaze) is stable since a moving gaze causes motion blur. Evolved animals use saccades when switching gaze direction to minimize the time during which the image is moving. During locomotion, compensatory movements of the eyes and head aim at minimizing the retina slip. The same issue is present when dealing with robots since most vision processing algorithms reach optimal performance with a stable image.

Head stabilization systems exist in the robotics literature, many of them being based on models of the vestibulo-ocular reflex observed in many vertebrates ([110]). These systems typically use a vestibular sensor (IMU, accelerometer etc.) as main sensory input to excite a leaky integrator. The remaining retinal slip (usually measured by optical flow) is then used to calibrate the gains of this integrator. Kawato's Feedback-Error-Learning model ([53]) is applied to the gaze stabilization problem in [101] where it is extended with a nonparametric regression network to improve the opto-kinetic response. In [63], the authors implement the Recurrent Decorrelation Control model [25] which forms a recurrent network with an artificial brainstem getting as input rotational speeds from the vestibular sensor, and an artificial cerebellum getting input from the brainstem and the retinal slip, and feeding back its output to the brainstem. A single neural network is used in [79] and excited directly by both the vestibular sensor output and the optical flow from the camera image to estimate the optimal compensatory motor command.

These systems reach very good performance but rely highly on the availability of a fast (typically around 500Hz) vestibular sensor in the head of the robot. Although this kind of sensors becomes more accessible, many robots still do not have an IMU in the head of the robot, but rather in the trunk. Very few approaches tackle the problem of head stabilization specifically during locomotion. The work in [95] relies on a forward kinematics and genetic algorithm to build an internal model of the head motion and compensate for it using a feedforward CPG based controller. This method however relies on offline optimization for the CPG parameters which has to be done for each different gait and is thus not very suitable for gaits changing in time (to cope with environmental specificities for instance).

In this chapter we propose a system for stabilizing the head of a legged robot during locomotion, which only relies on optical flow information. Assuming a periodic movement of the head (as is usually the case for legged locomotion), the system uses Adaptive Frequency Oscillators to learn the frequency and phase shift of the optical flow and generate compensatory movements to minimize the head motion. At convergence, the system is mostly feedforward and the feedback signal (the optical flow) is only used to finely tune the parameters of the oscillator. The system further shows the same interesting properties in terms of control as other oscillators (smooth modulation of parameters, resistance to perturbations etc.) This system is efficient even when using relatively slow cameras ($< 30Hz$) and is predictive in the sense that unlike reactive systems which use the last few sensor values to estimate the amplitude of the compensatory movement at the next step, our controller generates a compensatory signal which is phase locked with the optical flow signal. It effectively tries to predict the future, as the stabilizing commands are generated at a higher frequency than the optical flow. Our system is able to track changes in the movement applied to the robot and adapt its parameters to go back to a stabilized gaze. We show that our system can be used to stabilize the gaze of a moving robot using multiple degrees of freedom in the head. Further, it can be applied to tracking objects of arbitrary shape, colors and textures moving rhythmically.

In the following sections we present the system and its properties, then explain the influence of the different open parameters of the system. We explain how to use the system to stabilize the gaze using multiple degrees of freedoms in the head, and show that it can even stabilize the gaze of a robot on a moving object. We show that the system can be applied on legged locomotion (with the Hoap2 humanoid robot walking) and non legged locomotion (with a swimming salamander robot), as long as the movement of the head is periodic and close to a sine wave. We present the system applied on the real Hoap3 robot tracking a periodically moving apple, on the Pleurobot salamander robot walking and on a real human using a wireless pan-tilt camera.

3.2 Presentation of the system

In this section we present the details of the head stabilization controller. First we present a simplified version of the controller using a standard Hopf Adaptive Frequency Oscillator as first developed by Buchli, Righetti and Ijspeert ([19], [91]), and then show how we adapted it to satisfy the requirements of the head stabilization problem. Note that the AFOs are used here in a different manner as in the previously cited papers. Here we use the AFO in fully closed loop (the forcing signal changes the pattern generated by the oscillator and the oscillator's output influences the forcing signal, the optical flow). Furthermore, the goal here is not to learn the shape of the forcing signal as in [92], but to learn a correcting signal which leads to the suppression of the forcing

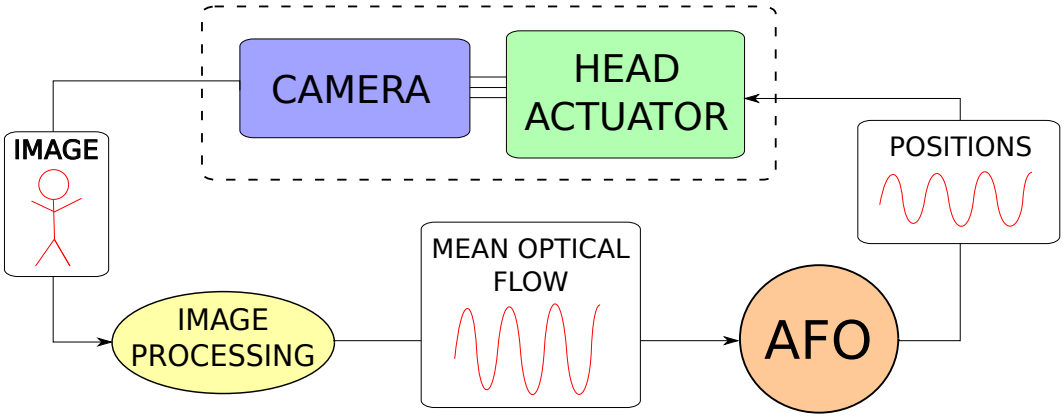


Figure 3.1: Outline of the architecture of the system for gaze stabilization

signal. In contrast, AFOs were previously always used in open-loop except in [19] where the AFO changes, very slowly, the frequency of the control and thus the teaching signal (only the frequency is changed in closed loop though, the amplitude remains constant).

Figure 3.1 outlines the architecture of the system. Images from the camera to stabilize are processed to obtain a measure of the optical flow using the standard OpenCV implementation of the Lucas Kanade - Shi Tomasi algorithm ([67], [100]). The optical flow signal is fed negatively to an Adaptive Frequency Oscillator which will tune its frequency, amplitude and phase shift so as to generate a signal phase locked with its teaching signal (in anti-phase with the optical flow), with the correct amplitude to minimize the optical flow. The output of the AFO is then used to control the head of the robot. We use here a slightly modified version of the Hopf AFO in polar coordinates in which we removed the influence of the forcing signal on the radius of the oscillator, to avoid divergence with high coupling terms.

The equations of the AFO are given below:

$$\dot{r} = \gamma(1 - r^2)r \tag{3.1}$$
$$\dot{\phi} = \omega - \sin\phi\beta F \tag{3.2}$$
$$\dot{\omega} = -\sin\phi\kappa F \tag{3.3}$$
$$x = r\cos\phi \tag{3.4}$$
$$\dot{\alpha} = -\eta x F \tag{3.5}$$
$$\theta = \alpha x + O \tag{3.6}$$

where r is the radius of the oscillator (i.e. the amplitude of its oscillations), ϕ its phase, ω its frequency and θ its output here used to control the position of the head actuator. α here directly defines the amplitude of the oscillations and O their offset. F is an external forcing signal (here the opposite of the mean optical flow). κ and β are scaling factors for the forcing signal. We describe the effect of these

scaling factors in Section 3.3. Equations 3.1 and 3.2 describe a limit cycle of radius 1. The forcing term in Equation 3.2 causes the phase to synchronize with that of the forcing signal (as in a standard forced Hopf oscillator), while a similar forcing on ω (Equation 3.3) tunes the frequency to that of the forcing signal. When the oscillations are synchronized (same frequency and same phase) with the forcing signal, the correlation between x and F is maximized and α starts increasing according to Equation 3.5, causing the head of the robot to oscillate in anti-phase with the optical flow with increasing amplitude, and thus decreasing the retinal slip, until the flow is about null. All the parameters of the generated compensatory signal are effectively learned such that they are conserved if the forcing term F is removed. This is particularly useful to deal with varying camera speeds, communication problems or occlusions.

Theoretically, this system works by itself. However, the convergence of the frequency is typically slow in the experiments by Buchli and Righetti (a few hundred seconds). This is mainly due to the fact that setting high values to β and κ changes a lot the shape of the oscillations of the AFO as well as issues discussed in Section 3.3. Figure 3.2 shows how the output of the oscillator is modified when β and κ are increased. When β and κ are high, the shape of the oscillations is highly modified from the original cosine wave. Furthermore, having too high coupling terms, when dealing with head stabilization, would cause divergence of the system. Indeed, since a jerky output as in Figure 3.2 (bottom) would cause a high optical flow which would in term induce a higher forcing etc.

To solve this problem and obtain fast convergence of the frequency while keeping control on the shape of the oscillations we used two phases for the AFO. The first phase, ϕ_1, is used only to learn the frequency of the forcing signal. The second phase ϕ_2 is the actual phase of the oscillations, and is coupled to the forcing signal for synchronization, with a different coupling term ϵ. Typically we set $\epsilon \ll \kappa$ so that the shape of the oscillations is not altered too much. These two phases share the same value for ω, so that the frequency learned is reflected on the oscillations of the head. Note that this system is equivalent to an AFO passing its frequency to a Hopf oscillator, and thus the proofs of convergence of AFOs in [91] remain valid and the properties of the Hopf oscillator are conserved.

The equations of the final system become :

$$\begin{aligned}
\dot{r} &= \gamma(1 - r^2)r && (3.7)\\
\dot{\phi_1} &= \omega - \sin\phi_1 \beta F && (3.8)\\
\dot{\phi_2} &= \omega - \sin\phi_2 \epsilon F && (3.9)\\
\dot{\omega} &= -\sin\phi_1 \kappa F && (3.10)\\
x &= r\cos\phi_2 && (3.11)\\
\dot{\alpha} &= -\eta x F && (3.12)\\
\theta &= \alpha x + O && (3.13)
\end{aligned}$$

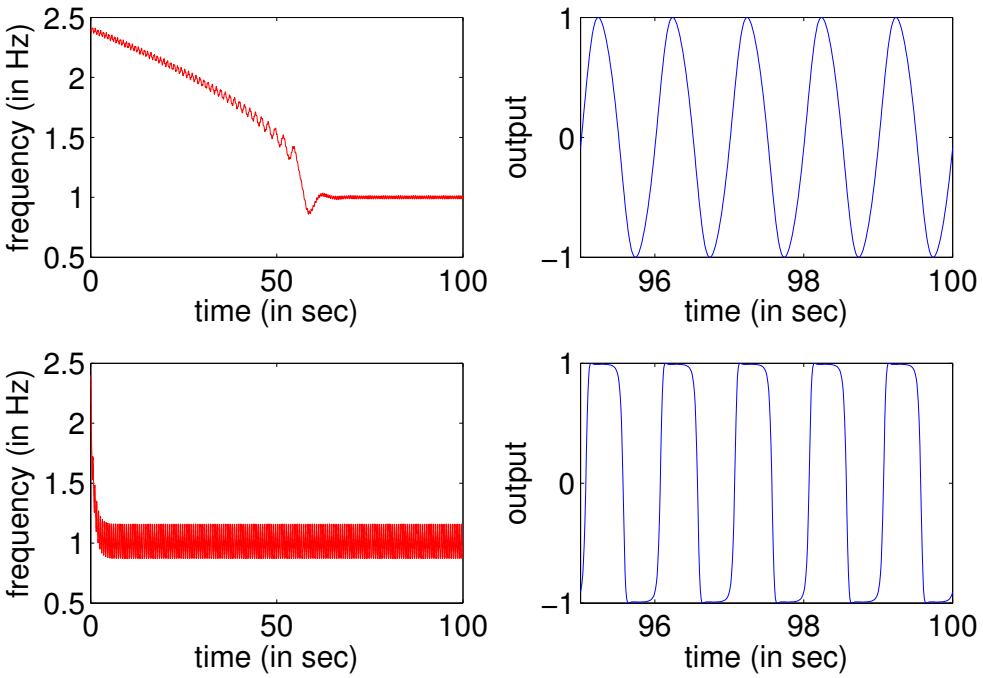


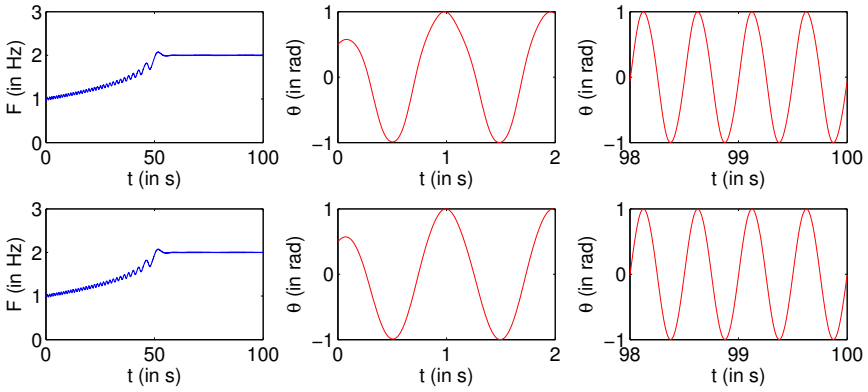
Figure 3.2: Evolution of the frequency (left) and shape of the output (right) for small scaling factors $\kappa = \beta = 2$ (top) and big scaling factors $\kappa = \beta = 50$ when the AFO is forced by the signal $F = sin(2\pi t)$.

Figure 3.3 shows a comparison of this modified AFO (Equations 3.7 to 3.13) with the original version (Equations 3.1 to 3.6), when forced by a 2Hz sine wave, and with slow, medium and fast convergence rates. Let us first note that the convergence behavior of both version of the AFO is exactly the same in all cases. This is quite obvious since the equations for ω and ϕ_1 are the same. When setting low convergence rates, both version of the AFO display similar behaviors, whether during the transient phase (first seconds, when the frequency has not converged yet) or after convergence (last seconds). When setting medium convergence rates, the original waveform is only slightly deformed compared to a perfect sine wave after convergence. However, during the transient phase, the waveform is significantly perturbed by the coupling with the forcing signal. The new version of the AFO in contrast is affected neither during the transient nor after convergence, due to the even smaller coupling strength ϵ. For high convergence rates this effect of the forcing on the output wave is even more visible. While the new AFO outputs a perfect sine wave, the output of the original version is very skewed.

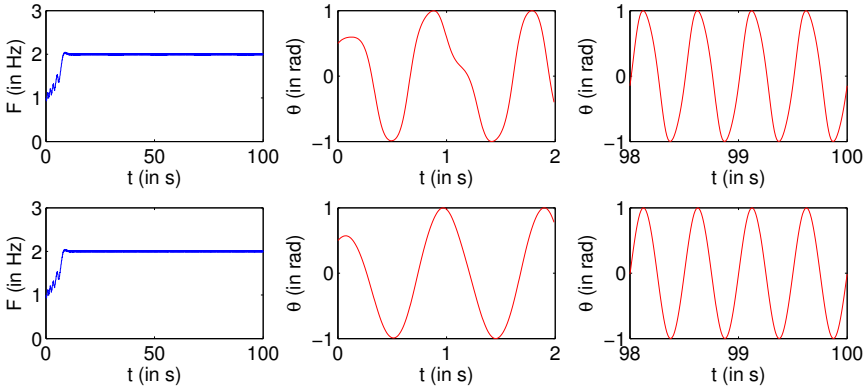
Another difference of the modified AFO compared to original one is that the speed of convergence of the frequency and the influence of the forcing on the output can be controlled independently. Figure 3.4 shows the behavior new AFO when setting a high value for ϵ while keeping the frequency convergence factors small. The result is a slow converging frequency while the output is highly perturbed by the forcing signal (note that ϵ is here so strong that the apparent frequency in the transient phase is already that of the forcing signal while the actual intrinsic one is still close to 1Hz). While this is an extreme example having most likely no concrete application, controlling the influence of the forcing signal on the output can have practical use for instance to force the outputs being in phase already from the start while the actual frequency is learned more slowly for increased stability (in the case of a noisy signal for instance).

3.3 Parameter tuning

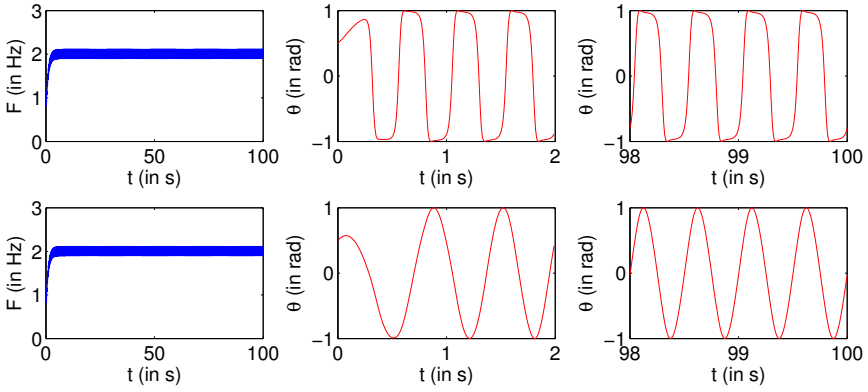
In this section we study the influence of the parameters κ and β on the convergence of the frequency of the system. Note that ϵ only acts on the second phase ϕ_2 which has no influence on the frequency modulation. Figure 3.5 shows the results of systematic tests monitoring the convergence time and the error after convergence for varying values of κ and β. The forcing signal used for this experiment was obtained by recording the optical flow when rotating a simulated camera in the air in a texturized scene around its pitch axis with a frequency of 2Hz (in the Webots robotics simulator [116]), and normalizing its amplitude. We used eight instances of our oscillator initialized at eight different frequencies uniformly distributed around the desired frequency.



(a) Small convergence rates ($\kappa = \beta = 2$). Top: original AFO, bottom: modified AFO



(b) Medium convergence rates ($\kappa = \beta = 5$). Top: original AFO, bottom: modified AFO



(c) Large convergence rates ($\kappa = \beta = 50$). Top: original AFO, bottom: modified AFO

Figure 3.3: Evolution of the frequency (left), shape of the output during the first seconds (middle column) and during the last seconds (right) for small convergence rates (a), medium convergence rates (b) and big convergence rates (c) when the AFO is forced by the signal $F = sin(4\pi t)$ (2Hz) and initialized with a frequency of 1Hz.

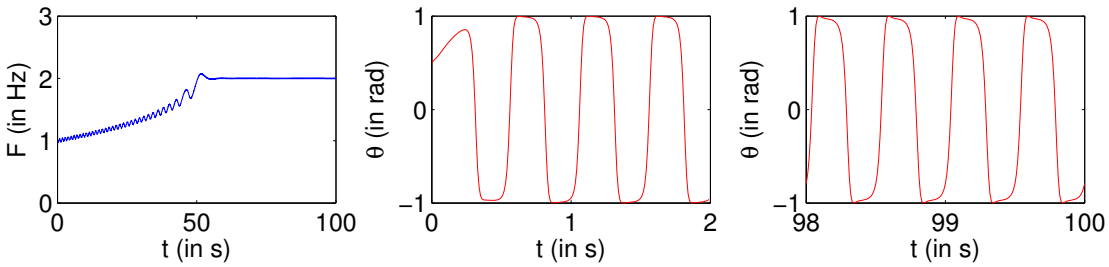


Figure 3.4: Evolution of the frequency (left), shape of the output during the first seconds (middle) and during the last seconds (right) for small convergence rates $\kappa = \beta = 2$ and a high value for $\epsilon = 50$ when the AFO is forced by the signal $F = sin(4\pi t)$ (2Hz) and initialized with a frequency of 1Hz.

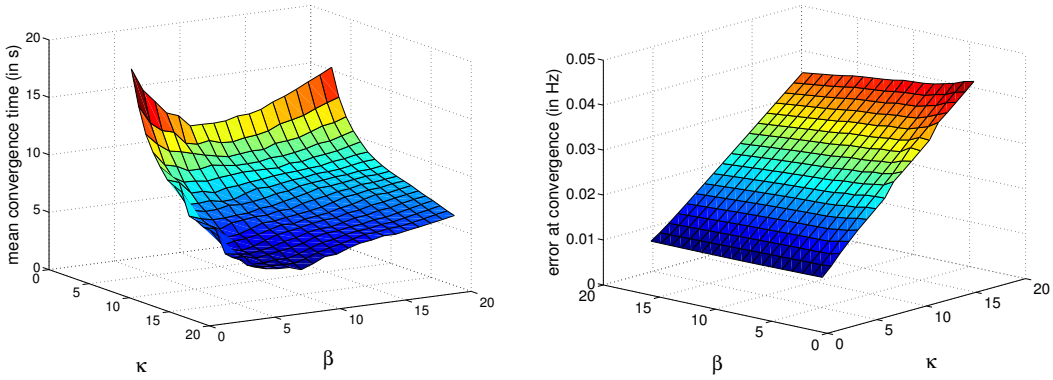


Figure 3.5: Convergence time $\mathcal{T}_c(S)$ (left) and error after convergence $\mathcal{E}_c(S)$ (right) for different values of κ and β when the system is forced by a normalized optical flow signal of frequency 2Hz.

The convergence time $\mathcal{T}_c(S)$ and error after convergence $\mathcal{E}_c(S)$ of a signal S (here the optical flow) to a desired value s are defined as follows:

$$\mathcal{T}_c(S) = \min(t), \forall t > \mathcal{T}_c(S), |S(t) - s| < \lambda \tag{3.14}$$

$$\mathcal{E}_c(S) = \frac{1}{\mathcal{T}_f(S) - \mathcal{T}_c(S)} \int_{\mathcal{T}_c(S)}^{\mathcal{T}_f(S)} |S(t) - s| dt \tag{3.15}$$

where $\mathcal{T}_f(S)$ is the final time of the signal S and λ is a chosen small value (in this study we used $\lambda = 0.25$). In clear, $\mathcal{T}_c(S)$ is defined as the minimum time after which the signal S stays bounded in a neighborhood of a desired value s, and $\mathcal{E}_c(S)$ as the mean of the instantaneous distance between $S(t)$ and s after $\mathcal{T}_c(S)$. These two quantities are then averaged over the eight oscillators.

The error after convergence $\mathcal{E}_c(S)$ (Figure 3.5, right) is basically proportional to κ, although it sightly decreases when β is increased for a given value of κ. The convergence

time $\mathcal{T}_c(S)$ (left) decreases monotonically with κ and meets a minimum for a specific value of β which depends on the value of κ. This minimum is however less visible when κ increases.

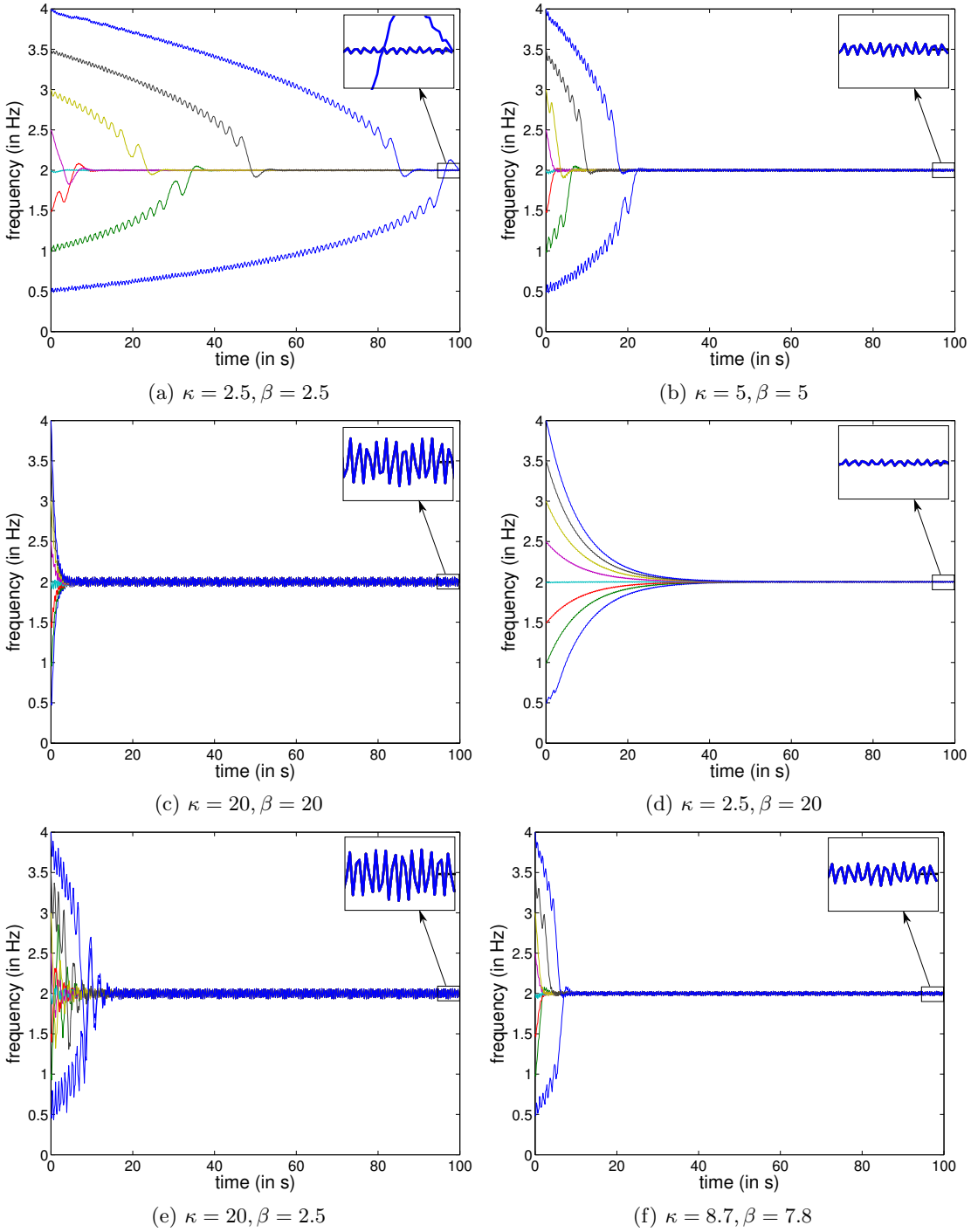
Figure 3.6 shows the evolution of the frequency of the system for characteristic values of κ and β and for different initial frequencies. For small values of κ and β (Figure 3.6a), the convergence takes a long time, especially for initial frequencies far away from the frequency of the forcing signal, while the remaining oscillations after convergence have very small amplitude. When β and κ are increased (Figures 3.6b and 3.6c), the convergence time decreases but the oscillations after convergence amplify. Increasing only β (Figure 3.6d) has a smoothing effect on the convergence. The AFOs with initial frequency far away from that of the forcing signal converge faster, while the others converge more slowly. Increasing κ while keeping β low (Figure 3.6e) causes the convergence to be very jerky, and increases the amplitude of the remaining oscillations at convergence compared to when both parameters are set high (Figure 3.6c). Figure 3.6f shows an example of a compromise between convergence speed and error after convergence.

This study will serve as a reference to choose the values of these parameters depending on whether convergence speed or precision at convergence is more critical, but also depending on whether we can have a good estimate of the frequency of the head movement (in which case we can afford to set lower values for κ while still converging fast enough). Note that only two parameters need to be tuned (η and ϵ can be fixed once and for all, they do not influence the convergence speed or quality). Also note that for any value of κ and β tested, the system converges, so the values of these two parameters is not too critical, but only define the quality of the stabilization. Typically during locomotion and especially for statically stable gaits, the frequency of the head motion is nearly that of the controlled robot motion, so one would want to initialize the oscillator frequency with this value. In Section 3.5 however, we show that in the case of the salamander robot swimming and the Hoap2 robot walking, this is not true for the pitch axis.

3.4 Extension to multiple axis stabilization

So far we have only considered one oscillator, for a single degree of freedom. However the system is fairly easy to extend to multiple degrees of freedom for the head. Typically one would use one AFO per degree of freedom. The only constraint here is finding the right forcing signal for each AFO.
To result in a successful head stabilization, the forcing signal for one degree of freedom should:

- have the same frequency as the motion of the robot around this axis.
- decrease towards zero when the head is stabilized around this axis.



(a) $\kappa = 2.5, \beta = 2.5$

(b) $\kappa = 5, \beta = 5$

(c) $\kappa = 20, \beta = 20$

(d) $\kappa = 2.5, \beta = 20$

(e) $\kappa = 20, \beta = 2.5$

(f) $\kappa = 8.7, \beta = 7.8$

Figure 3.6: Evolution of the frequency of the system for typical values of κ and β when the system is forced by a normalized optical flow signal of frequency 2Hz and for AFOs initialized with various initial frequencies.

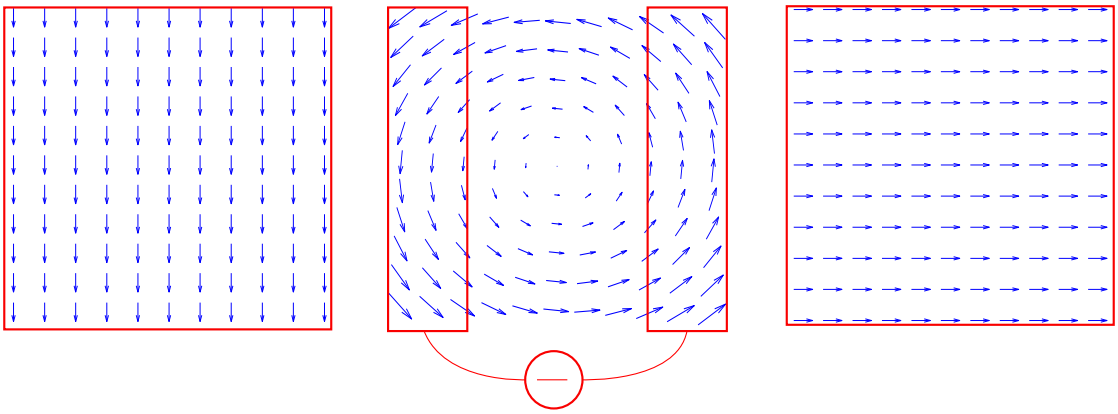


Figure 3.7: Typical patterns of optical flow for each axis, and the different parts of the optical flow used for each axis. Left: pitch axis: the vertical component of the flow is averaged over the whole window. Middle: roll axis: the vertical component of the flow is averaged in the left and right quarters of the image, and these averages are subtracted. Right: yaw axis: the horizontal component of the flow is averaged in the whole window

- have zero mean.

Note however that the forcing signal does not need to be an estimate of the head rotation speed around the considered axis in any way. In this chapter we typically use the optical flow since it is the most basic information provided by a camera, but we could just as well use for instance the position of an object around the center of the image, or the position and orientation of the horizon.

To extend the system to three axis stabilization (pitch, roll, yaw, as commonly defined in aviation), we use the following forcing terms for the corresponding AFOs:

- For the pitch: the mean of the y coordinate of the flow vectors of the whole image.
- For the yaw: the mean of the x coordinate of the flow vectors of the whole image.
- For the roll: the difference between the mean of the y coordinate of the flow vectors of the left quarter and the right quarter of the image.

Figure 3.7 shows the different parts of the optical flow used as forcing for each degree of freedom.

These three forcing terms are applied negatively to the AFOs, so that at convergence, the oscillators are in anti-phase with the optical flow. The following equations formalize the forcing for the pitch, roll and yaw axis (respectively F_p, F_r and F_y)

$$F_p = -\frac{1}{K}\sum_{i=1}^{m}\sum_{j=1}^{n}\mathcal{F}_{ij}^{y} \tag{3.16}$$

$$F_y = -\frac{1}{K}\sum_{i=1}^{m}\sum_{j=1}^{n}\mathcal{F}_{ij}^{x} \tag{3.17}$$

$$F_r = -\left(\frac{1}{K_l}\sum_{i=1}^{m}\sum_{j=1}^{\frac{n}{4}}\mathcal{F}_{ij}^{y} - \frac{1}{K_r}\sum_{i=1}^{m}\sum_{j=\frac{3n}{4}}^{n}\mathcal{F}_{ij}^{y}\right) \tag{3.18}$$

where K is the number of non zero flow vectors in the whole image, K_l and K_r are the numbers of non zero flow vectors in the left and right quarters of the image, m and n are the dimensions of the image, and F_{ij}^{x} and F_{ij}^{y} are the x and y components of the optical flow vector computed at position (i, j).

Note that these three forcing terms do not give a direct measure of the rotation speed of the head around each axis. This is not needed by our system. The forcing terms used for each axis need however to satisfy the three conditions given earlier. In our case, this implies that the pitch axis of the head moves the image approximately along its y axis, the yaw along its x axis, and that the roll rotates the image around its center. In the case of a head with two cameras on each side for instance, the forcing for the roll axis F_r may not work as it is. It could be adapted by taking the difference of the flow of the left part of the left camera image and the right part of the right camera image, for instance

Also note that if the amplitude of the motion around one axis is significantly larger than around the others, it may initially hide the smaller motions in the flow field. For instance a large pitching motion may hide a smaller rolling motion. However, in this case, after the AFO responsible for stabilizing the pitch angle has converged, this large pitching (vertical) flow will disappear and the rolling motion will be revealed. The AFO responsible for the rolling motion will then be able to stabilize the camera around this axis.

3.5 Results

3.5.1 Results in simulation

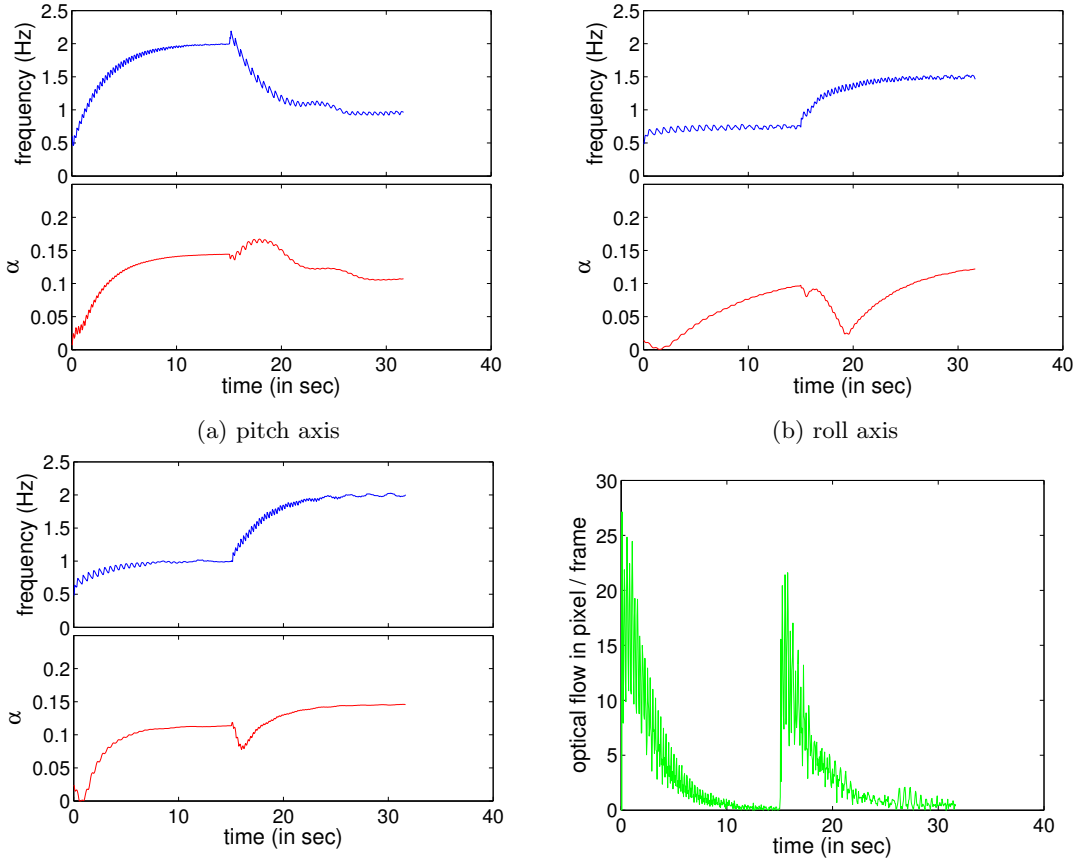
In this section we present results of the system actually applied to the head stabilization problem. All the experiments described below have been carried out using Webots [116], a physics based simulator for robotics. Here we only actuate the head of the robot (not the eyes) but applying it to the eyes also should be straightforward. The camera

is a simulated pinhole camera with a field of view of 45° and providing an image of 320 x 240 pixels at 20Hz. (which is below standard for robotics cameras). The optical flow is computed from the camera images the same way in simulation and on the real robot, and is thus subject to noise (image noise, processing artifacts etc.). For all the following experiments, to show the learning properties of our system, the AFOs were initialized to a frequency different to that of the forcing signal. However, in a real scenario, if the frequency of the locomotion is known, one could initialize the AFOs to that of the walking. This would improve the convergence speed. The reader is advised to refer to the video at [4] for a better insight into the following experiments.

Figure 3.8 shows the evolution of the frequency and the amplitude of the system when a robot (here the Fujitsu Hoap2 humanoid robot) is rotated in the air with sine waves of different frequencies for the pitch, roll and yaw axis (see Figure 3.9d). One instance of our oscillator is used per degree of freedom with different forcing signals as explained in Section 3.4. To demonstrate the self tuning ability of the system, the frequency of the motion for the pitch axis is set arbitrarily to 2Hz, for the roll 0.75Hz and for the yaw 1Hz. At $t = 15s$, the frequencies are switched to: pitch axis: 1Hz, roll axis: 1.5Hz, yaw axis: 2Hz. The AFO is initialized with a frequency of 0.5 Hz.
The frequency of the AFOs controlling each actuator of the head quickly converges to that of the motions applied to the robot and the amplitude starts increasing until the optical flow is minimum. When the frequencies of motion are suddenly altered, the system tracks the change of frequency and recovers until the optical flow is minimal again. The resulting flow after convergence is reduced to less than 1 pixel/frame both times, in about 10 seconds.

Our system does not assume that the movement to compensate is a rotation. It actually works even for pure translations. Figure 3.9 describes a similar experiment as the previous one, but this time with the robot periodically translated along the x and y axis (the y axis here is the vertical, while the x axis is sideways with respect to the robot) with sine waves of different frequencies: for the x axis 1Hz, for the y axis: 2Hz. At $t = 10s$, the frequencies are switched to 2Hz for the x axis and 1Hz for the y axis. Again the system converges quickly leading to lateral and vertical head movements that almost completely suppress the optical flow. After the switch in frequency, the system recovers and goes back to nearly perfect stabilization.

As explained in Section 3.2, our system generates oscillations whose shape can be slightly modified by the forcing signal, but remains close to a sine. Figure 3.10 shows the behavior of the system when the robot is rotated around its pitch axis with waves of different shapes. For every shape, the system manages to learn the main frequency of the optical flow signal. It also manages to reduce the optical flow, leading to a more stabilized gaze than without the system. However, the further the shape of the rotation is from a sine, the worst the performance, as expected.



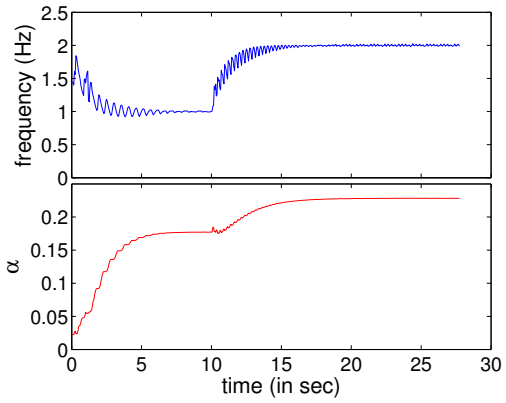
(a) pitch axis

(b) roll axis

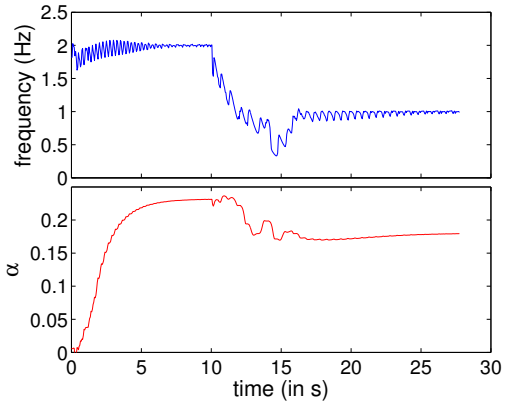
(c) yaw axis

(d) optical flow

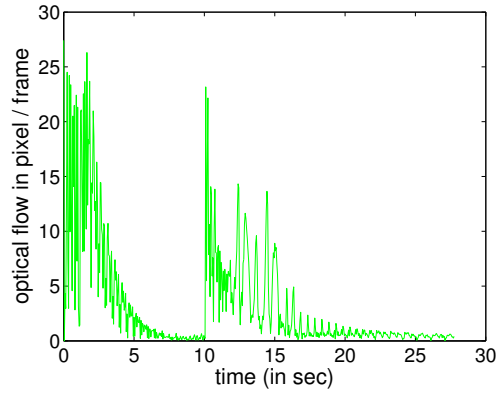
Figure 3.8: Evolution of the frequency and the amplitude (α) of the oscillator when the robot is rotated in the air around the three axis pitch, roll and yaw, by sine waves of different frequencies. For the pitch axis: 2Hz, for the roll axis: 0.75Hz and for the yaw axis: 1Hz. At $t = 15s$ the frequencies are switched to: pitch axis: 1Hz, roll axis: 1.5Hz, yaw axis: 2Hz. The AFO is initialized with a frequency of 0.5 Hz. Figure 3.8d shows the evolution of the norm of the mean optical flow vector over time.



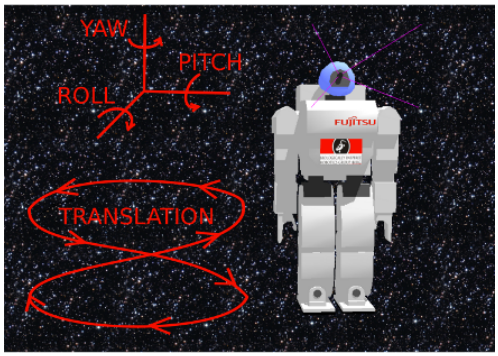
(a) x axis



(b) y axis



(c) optical flow



(d) scene

Figure 3.9: Evolution of the frequency and the amplitude (α) of the oscillator when the robot is translated in the air along the x and y axis by sine waves of different frequencies. For the x axis: 1Hz and for the y axis: 2Hz. At $t = 10s$ the frequencies are switched to: x axis: 2Hz, y axis: 1Hz. The AFO is initialized with a frequency of 2 Hz. Figure 3.9c shows the evolution of the norm of the mean optical flow vector over time.

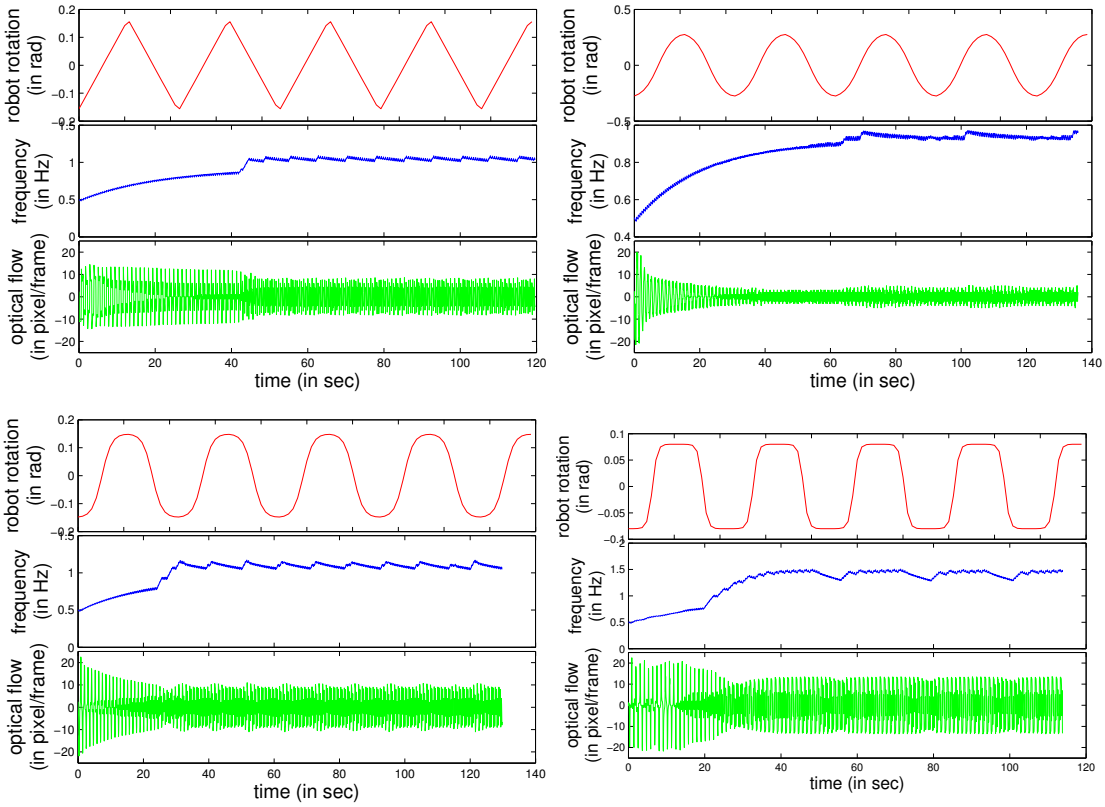


Figure 3.10: Evolution of the frequency (middle) of the head stabilizing oscillator and the optical flow (bottom) when the robot is rotated around its pitch axis with waves of different shapes (top), from near triangle to nearly step functions.

Figure 3.11 shows the performance of the system when the robot is rotated around its pitch axis with a chirp ($\sin(2\pi(\omega_0 + kt)t)$), first with a relatively slow changing frequency, and then with a much faster changing one. When the frequency of the robot rotation is changing slowly, the system is able to track these changes fast enough to enable good gaze stabilization. When the frequency of the movement is changing faster, the system still tracks it but not fast enough to lead to optimal performance stabilization. Figure 3.11 also shows the actual range of effectiveness of the system. For both cases, at frequencies higher than 5.5Hz the system is not able find the frequency of the teaching signal and the stabilization does not work anymore. However, it is important to note that this limit is not intrinsic to Adaptive Frequency Oscillators, which have an infinite basin of attraction. This limit is simply due to the low sampling rate of the optical flow, causing the signal to noise ratio to be very high at 5.5Hz .

Figure 3.12 shows the behavior of the system in the presence of external perturbations. The robot is successively perturbed by applying random rotations and translation for a short period of time (0.2s at $t = 10s$) and then for a longer period (2s at $t = 15s$). When the perturbation is short, the frequency and amplitude of the oscillator hardly change at all, and the stabilization recovers quickly. When the perturbation lasts longer, the frequency deviates and the amplitude drops dramatically. When the perturbation stops, the system relearns the frequency of the optical flow and re-stabilizes the head.

Our system relying only on visual cues, it can also be used to stabilize the gaze of the robot on periodically moving objects of arbitrary shapes, colors etc. Figure 3.13 shows results of the system applied to the Hoap2 robot tracking a sphere (the moon) being translated with a sine wave along the x and y axis (vertical and sideways). The robot is not moved in this experiment. The frequencies of the motion of the sphere along the x and y axis are set respectively to 2Hz and 1Hz. At $t = 15s$, the frequencies are switched to 1Hz for the x axis and 2Hz for the y axis. The system is able to stabilize the gaze of the robot on the object almost perfectly, and tracks changes in the movement of the object. The result is the object staying almost exactly in the center of the image after convergence (about 5s).

We now show the system applied to robot locomotion. Figure 3.14 shows the evolution of the frequency and the amplitude of the oscillators controlling the pitch, roll and yaw axis of the camera attached at the tip of the head of a simulated salamander robot swimming. The salamander robot ([49]) is a modular 12DOF robot controlled with coupled oscillators (central pattern generators), and is capable to switch from walking to swimming. For this experiment, it is swimming by generating a traveling wave along its body whose frequency and amplitude can be modulated. The frequency of this wave is initially set to 1Hz. At $t = 30s$, the frequency is switched to 1.5Hz. The frequency of the oscillators is initialized to 0.5Hz. Again the system successfully stabilizes the gaze of the robot along the two axis, and tracks the change of frequency of the motion. The remaining optical flow after convergence is due partially to the

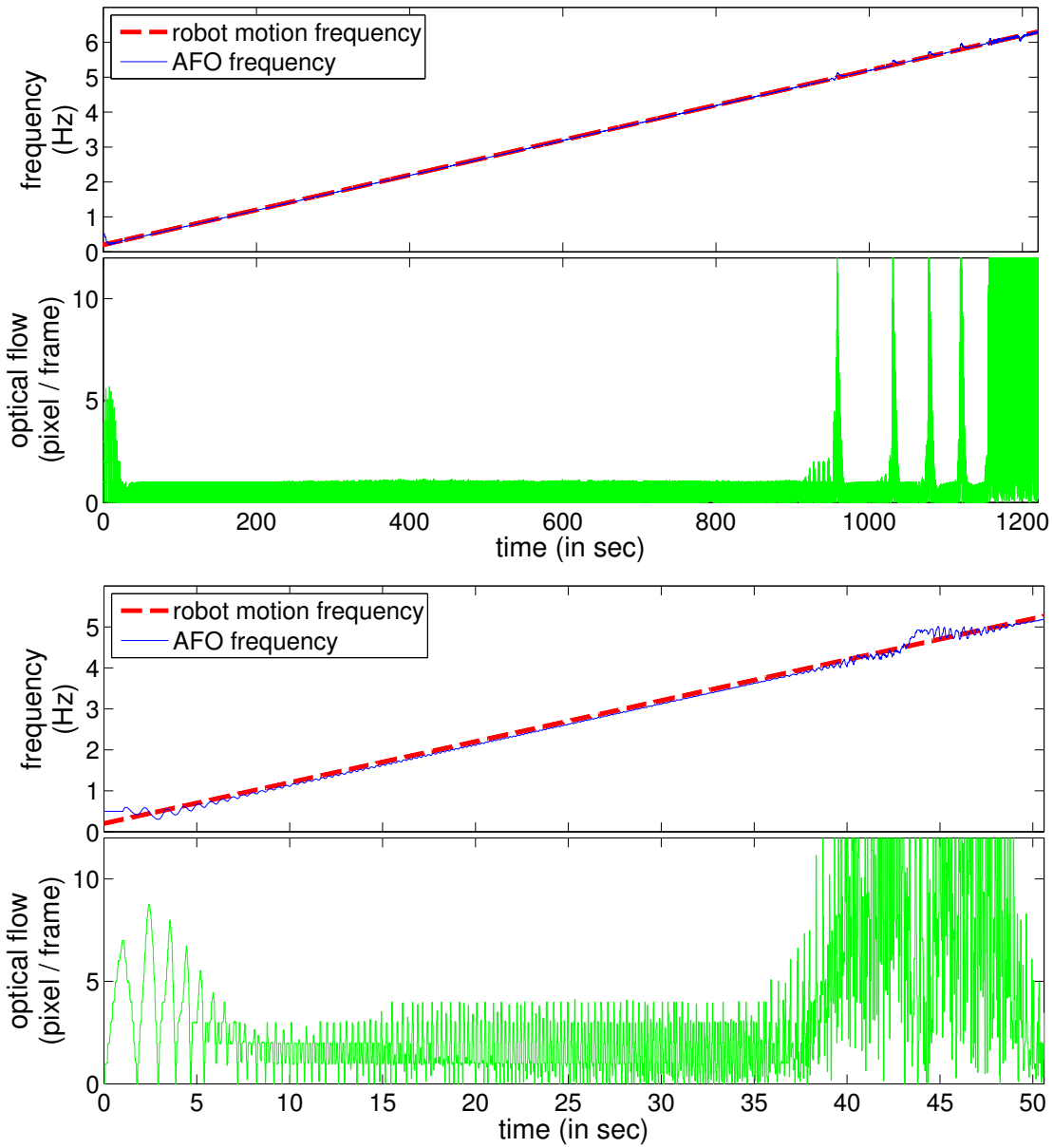


Figure 3.11: Evolution of the frequency of the head stabilizing oscillator and the optical flow when the robot is rotated around its pitch axis with a a sine wave with frequency increasing in time. Top: slow changing frequency, Bottom: fast changing frequency.

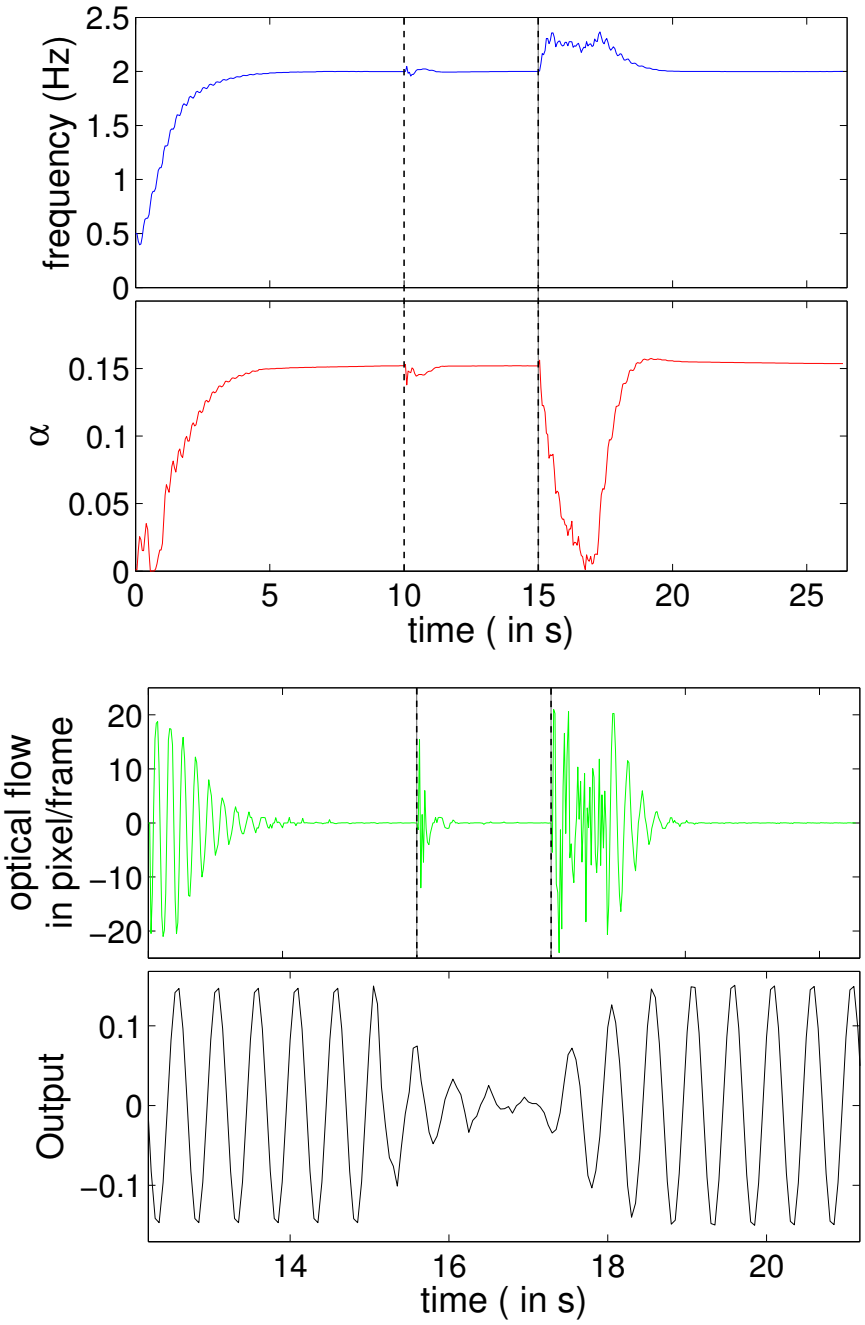
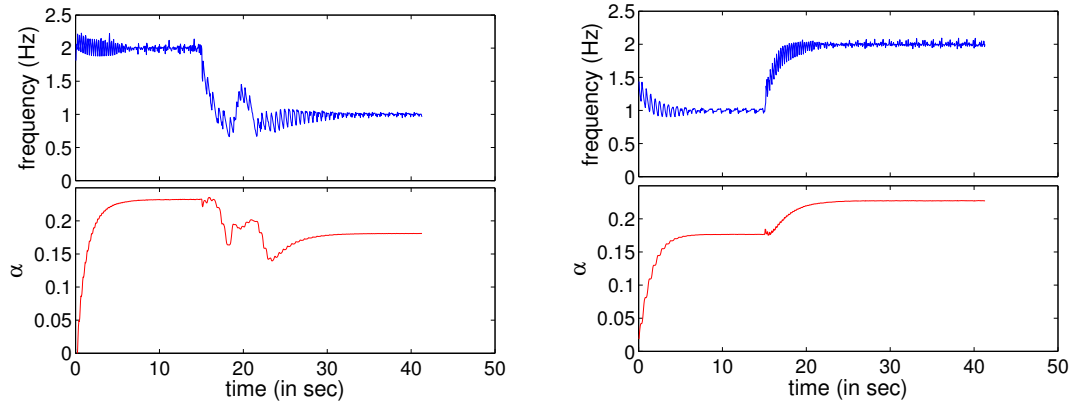
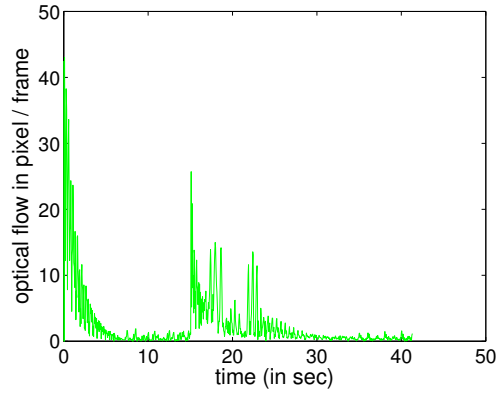


Figure 3.12: Evolution of the frequency and amplitude of the head stabilizing oscillator (left) and the optical flow (right) when the robot is rotated around its pitch axis with a a sine wave of frequency 2Hz. At $t = 10s$ random translation and rotation are applied to the robot for 0.2s and at t=15 for 2 seconds

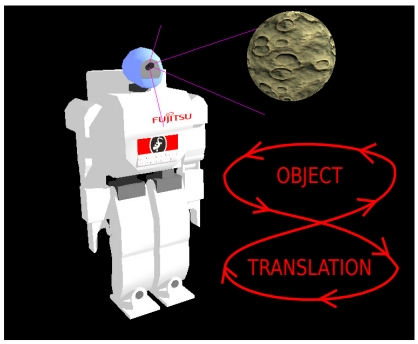


(a) x axis

(b) y axis



(c) optical flow



(d) scene

Figure 3.13: Evolution of the frequency and the amplitude (α) of the oscillator when an object (here a sphere) is translated in the air along the x and y axis by sine waves of different frequencies. For the x axis: 2Hz and for the y axis: 1Hz. At $t = 15s$ the frequencies are switched to: x axis: 1Hz, y axis: 2Hz. The AFO is initialized with a frequency of 2 Hz. Figure 3.13c shows the evolution of the norm of the mean optical flow vector over time.

forward motion of the robot, as shown in Figure 3.14d.

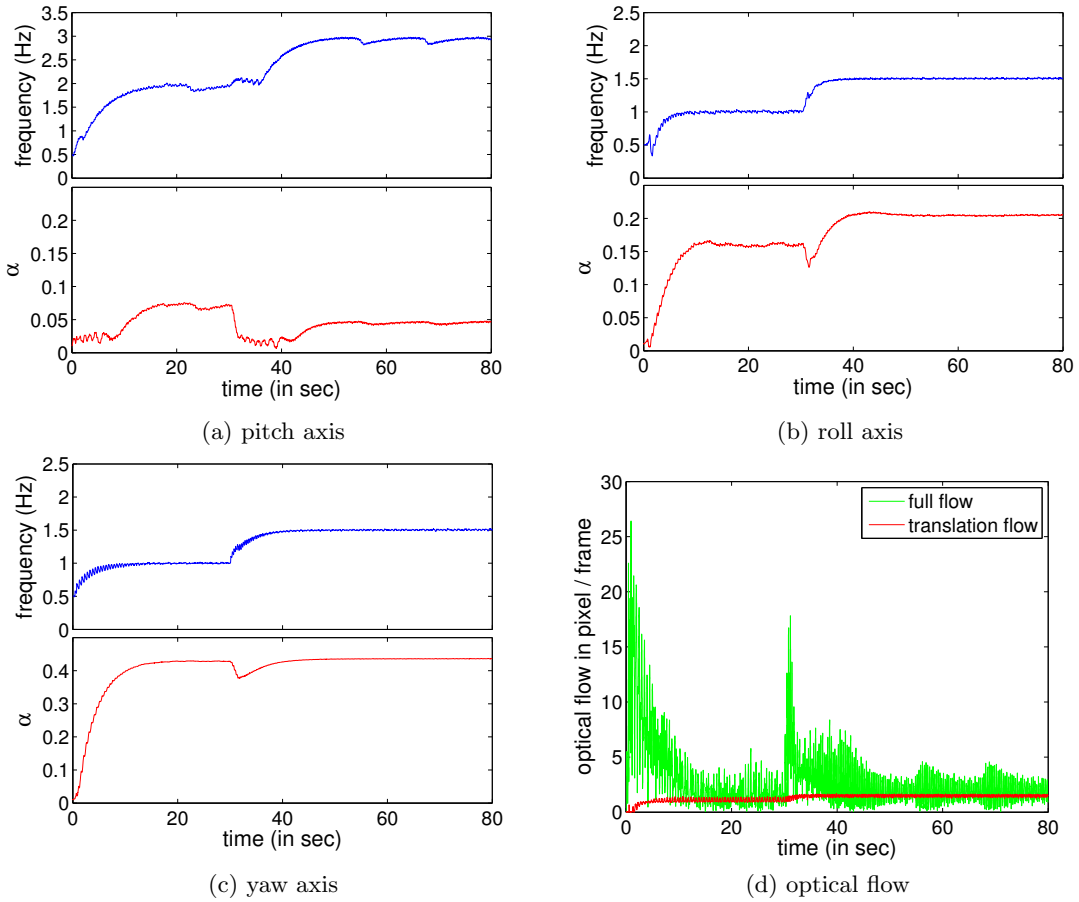
In the case of the salamander swimming, we could have initialized the frequency of the head stabilizing oscillator to the frequency of the motion control (we did not to demonstrate the tuning abilities of the system). Note however that the frequency of the motion of the head around the pitch axis is twice that of the general motion of the robot. The head is diving in the water at each half period of the traveling wave controlling the robot. This particularity is highly related to the gait used here and is very difficult to predict a priori (it would need complex modeling of the fluid dynamics). Our system however learns the correct frequency for this axis without the need of any modeling. Figure 3.15 shows snapshots of the salamander swimming, with and without the head stabilization system enabled.

Figure 3.16 shows a similar experiment as above with the Hoap2 walking. Three axis stabilization is used in the same way as for the other experiments. The robot is controlled using the default gait provided by Fujitsu. The frequency of the motion is not altered for this experiment, since the gait is only stable with the precomputed parameters. The frequency of the AFOs is initialized to 2Hz (different from that of the motion). Figure 3.16e shows the shape of the robot motion at the base of the head, for each axis. Even though this motion is quite far from a sine wave, the parameters of the system converge and gaze stabilization reaches decent performance, with the optical flow after convergence reduced to less than 7 pixels/frame. Note that, as in the case of the salamander robot swimming, the frequency of the motion around the pitch axis is about double that of the other axes.

3.5.2 Results with real hardware

In this section we present results of using our system with real hardware.

We performed the object tracking experiment with the real Hoap3 robot, which has embedded cameras in its head. An apple was attached to a spring, allowing it to swing horizontally and vertically, with different frequencies. We used here the exact same system as in the experiments in simulation. Taking into account the frame rate of the camera, the computation time of the optical flow and communication delays, we can provide our oscillator with visual forcing at a frequency of about 10Hz. Figure 3.17 shows the evolution of the frequency and amplitude of the two axis controlling the head. Here, the optical flow was not a good measure of the performance of the system, due to the high noise even after stabilization (see video [4]). Instead we used simple blob tracking to compute the position of the apple in the image frame (Figure 3.17c). Even with such a slow and noisy optical flow, the system is able to stabilize the object around the center of the image. Around $t = 45s$, the stabilization around the yaw axis gets worse for a couple of seconds, but quickly recovers. Note that the frequency of the apple motion is not perfectly constant here due to the natural damping of the spring



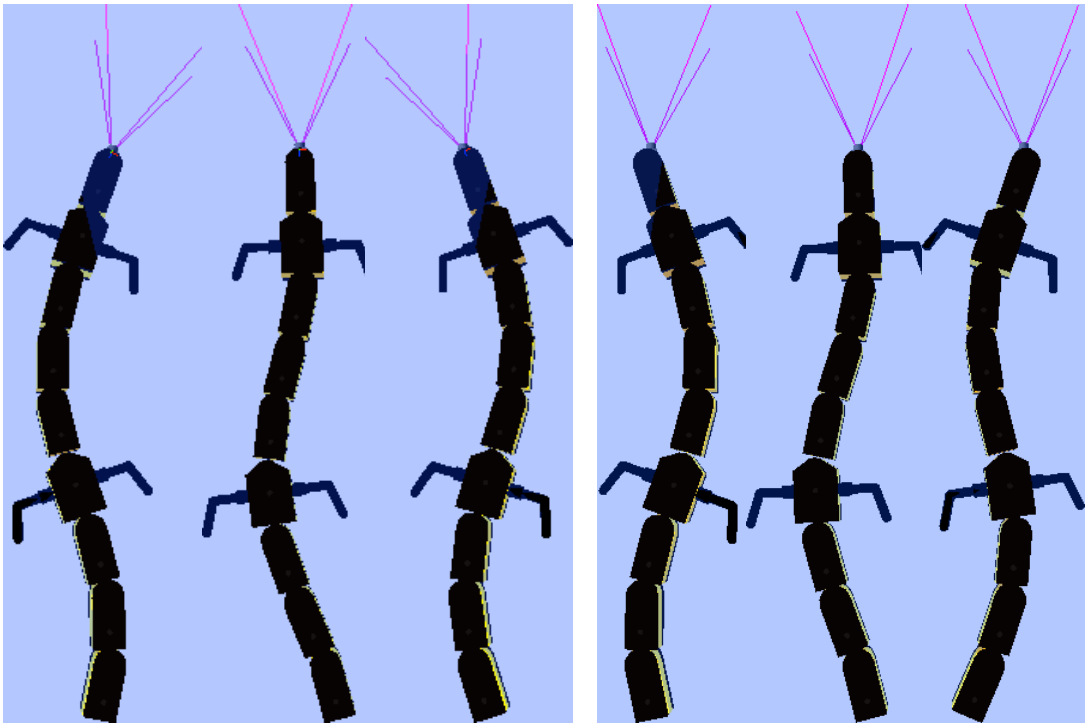
(a) pitch axis

(b) roll axis

(c) yaw axis

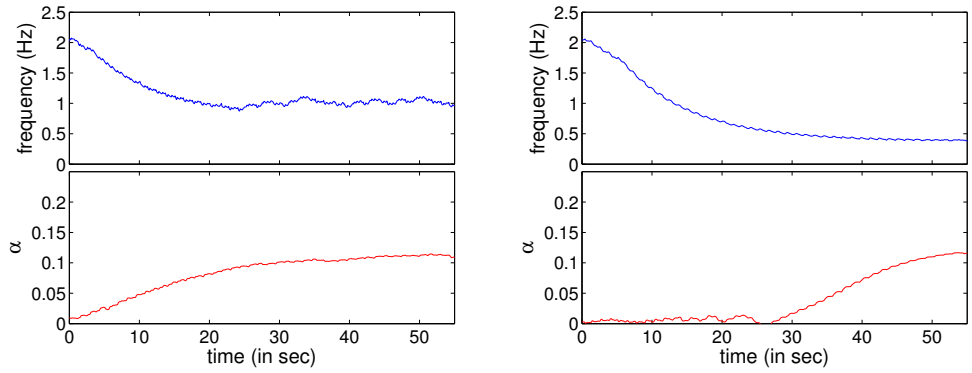
(d) optical flow

Figure 3.14: Evolution of the frequency and the amplitude (α) of the oscillator when the salamander robot swimming. The frequency of swimming is initially 1Hz and at $t = 30s$ the frequency is switched to 1.5Hz. The AFO is initialized with a frequency of 0.5Hz. Figure 3.14d shows the evolution of the norm of the mean optical flow vector over time as well as the flow due to the forward motion of the robot



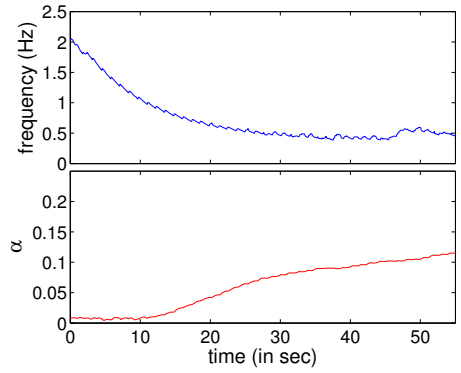
(a) (b) (c) (d) (e) (f)

Figure 3.15: Snapshots of the salamander robot swimming without gaze stabilization (a, b, and c) and with gaze stabilization (d, e and f). When the camera stabilization system is enabled, the gaze (highlighted by the purple camera frustums) always points in the direction of motion.

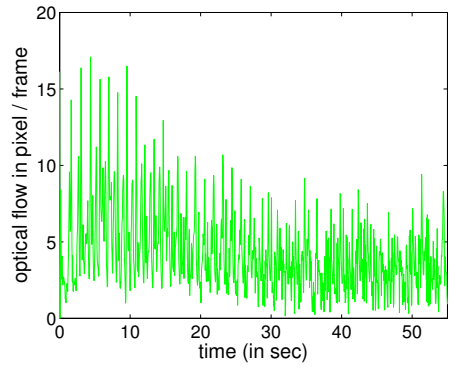


(a) pitch axis

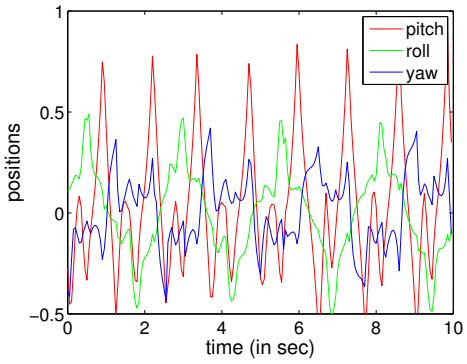
(b) roll axis



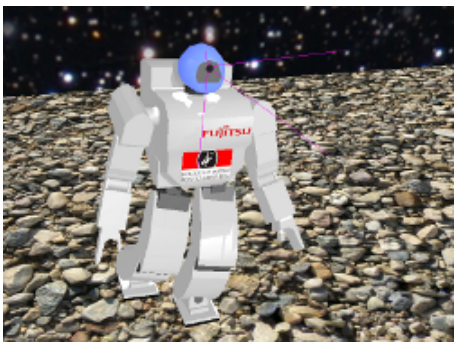
(c) yaw axis



(d) optical flow

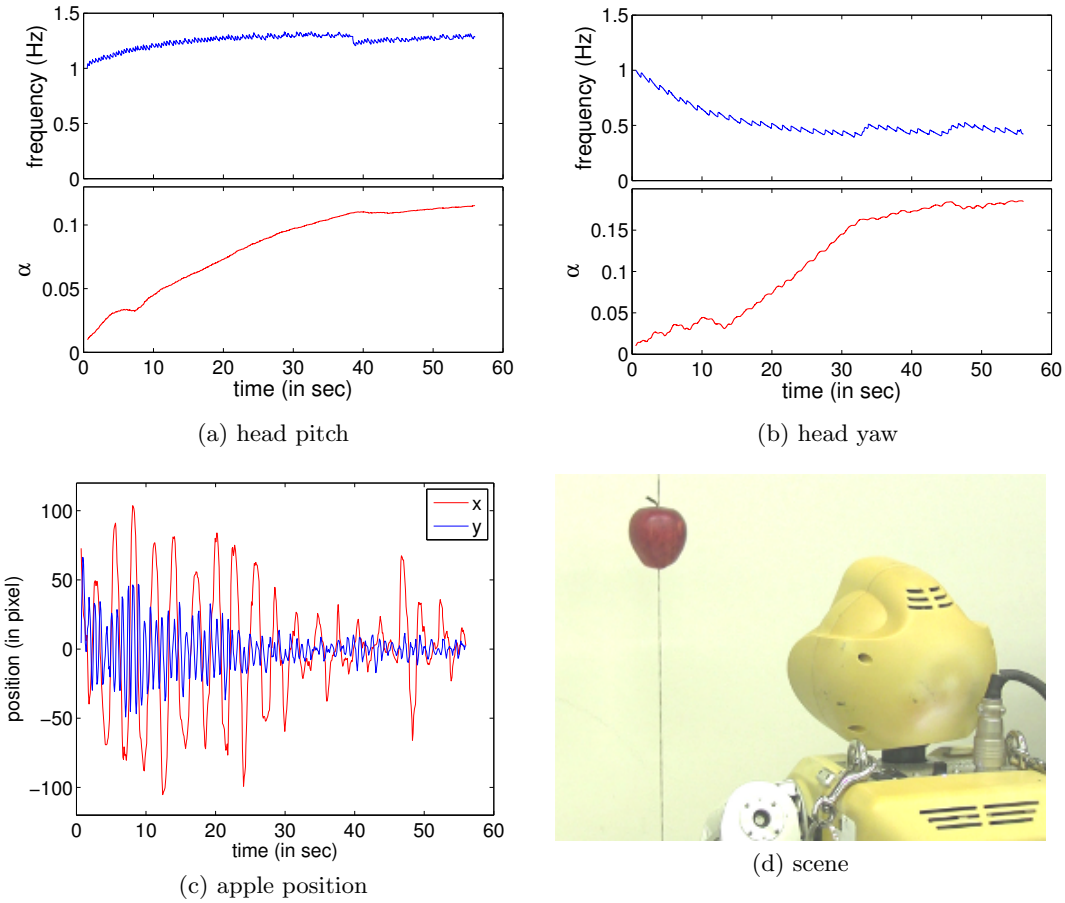


(e) head base rotation



(f) optical flow

Figure 3.16: Evolution of the frequency and the amplitude (α) of the oscillator when the Hoap2 robot is walking. The AFO is initialized with a frequency of 2Hz. Figure 3.16d shows the evolution of the norm of the mean optical flow vector over time. Figure 3.16e shows the shape of the rotation speed of the base of the head.



(a) head pitch

(b) head yaw

(c) apple position

(d) scene

Figure 3.17: Evolution of the frequency and the amplitude (α) of the oscillator when an apple is oscillating in front of the real Hoap3 robot. The apple is attached to a spring allowing it to swing horizontally and vertically.

and the air friction.

For the next experiments, we designed a pan-tilt wireless camera which can be attached either to a robot or a human. Figure 3.18 shows the whole structure of this camera. The device is composed of a wireless analog camera which images can be recovered from an external computer using an analog to digital converter. A digital webcam could also have been used but they typically display more motion blur than analog ones. A more dedicated camera should be considered in the future.

The camera is attached to two orthogonal servos, controlling its yaw and pitch angles. These servos, the Pololu DSM44, are capable of high speeds ($0.07sec/60°$), relatively high torques (1.6kg.cm) while only measuring 2.7cm and weighting 6g. They are controlled by a Pololu Micro Maestro servo controller which inputs/outputs are streamed through a pair of Pololu Wixel 2.4 GHz radio wireless controllers. The servos,

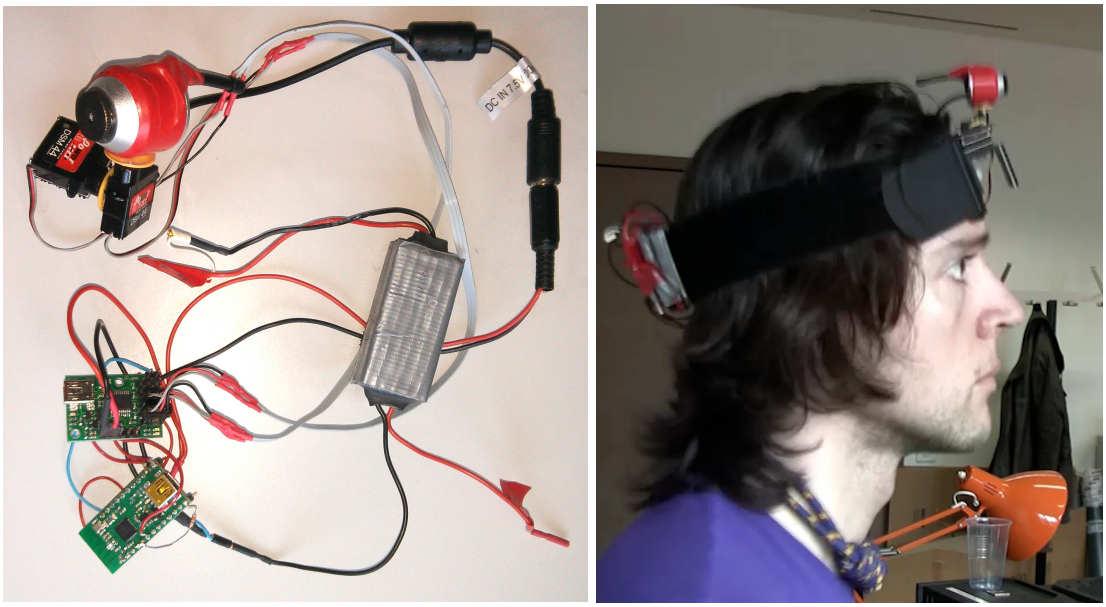


Figure 3.18: The designed actuated camera: a wireless camera is attached to two servos for pitch and yaw. These servos are controlled by a Pololu servo controller, and commands are exchanged wirelessly between the controller and an external PC with a wireless 2.4Gz radio card. The whole device is powered by batteries. Right, the device in action attached to my head.

controllers and camera are battery powered so that the whole device is completely wireless. Both image processing and servo trajectory computation are performed on an external PC. With a more powerful micro-controller, a the system could be made fully autonomous by implementing the AFO and the optical flow computation directly on board.

We performed tests with a salamander like robot, the Pleurobot, walking in the corridors of the laboratory as well as outside (i.e. non specific scenes). The Pleurobot gait applies a C-shape with alternating directions to the spine of the robot when walking, much like real salamanders. The down-side of this gait is that the head is moving a lot around the yaw axis. We attached our camera to the head of the Pleurobot and let the system stabilize itself. After convergence, the center of the corridor was maintained in the middle of the field of view throughout the gait cycle. Figure 3.19 shows snapshots of the camera images before and after convergence.

This experiment constitutes a first proof of concept, but the performance of the stabilization is not great. The scene is moving less after convergence than before but the image is not very still. This is mostly due to the fact that the gait of the Pleurobot is not really periodic due to slippage and tilting of the robot, and that the motion applied is quite far from a sine wave (abrupt left-right movements). Testing our system with a swimming gait for instance should greatly improve the performance.

We also performed tests attaching the camera to a human subject (myself) running on spot. The camera was attached on my head and I was running in my office in front of a plant (i.e. non specific scene). After convergence, the motion of the whole scene was reduced by a great amount. Figure 3.20 shows snapshots of the camera images before and after convergence. In this case the controller was more successful than with the Pleurobot to stabilize the camera. After convergence the plant in the scene stays roughly steady in the image while the background actually seems to rotate around it. Figure 3.21 shows the vertical position of my head during running. It is quite close to a sine wave, which explains the good performance of the stabilization controller.

These experiments show one of the great strength of our approach: the fact that the camera motion controller is decoupled from the robot motion controller. Thus our device can be attached seamlessly to any robot or even a human.

3.6 Conclusion

In this chapter, we presented a novel approach to the head stabilization problem during periodic movements. This specifically addresses question E (see Section 1.5 of the introduction of this thesis). Our system uses only visual cues, here optical flow, to stabilize the head of a robot subject to periodic motion, typically during locomotion. The system tries to predict the motion of the robot, by learning the frequency, phase and amplitude of the optical flow. All the learning is done online, and embedded

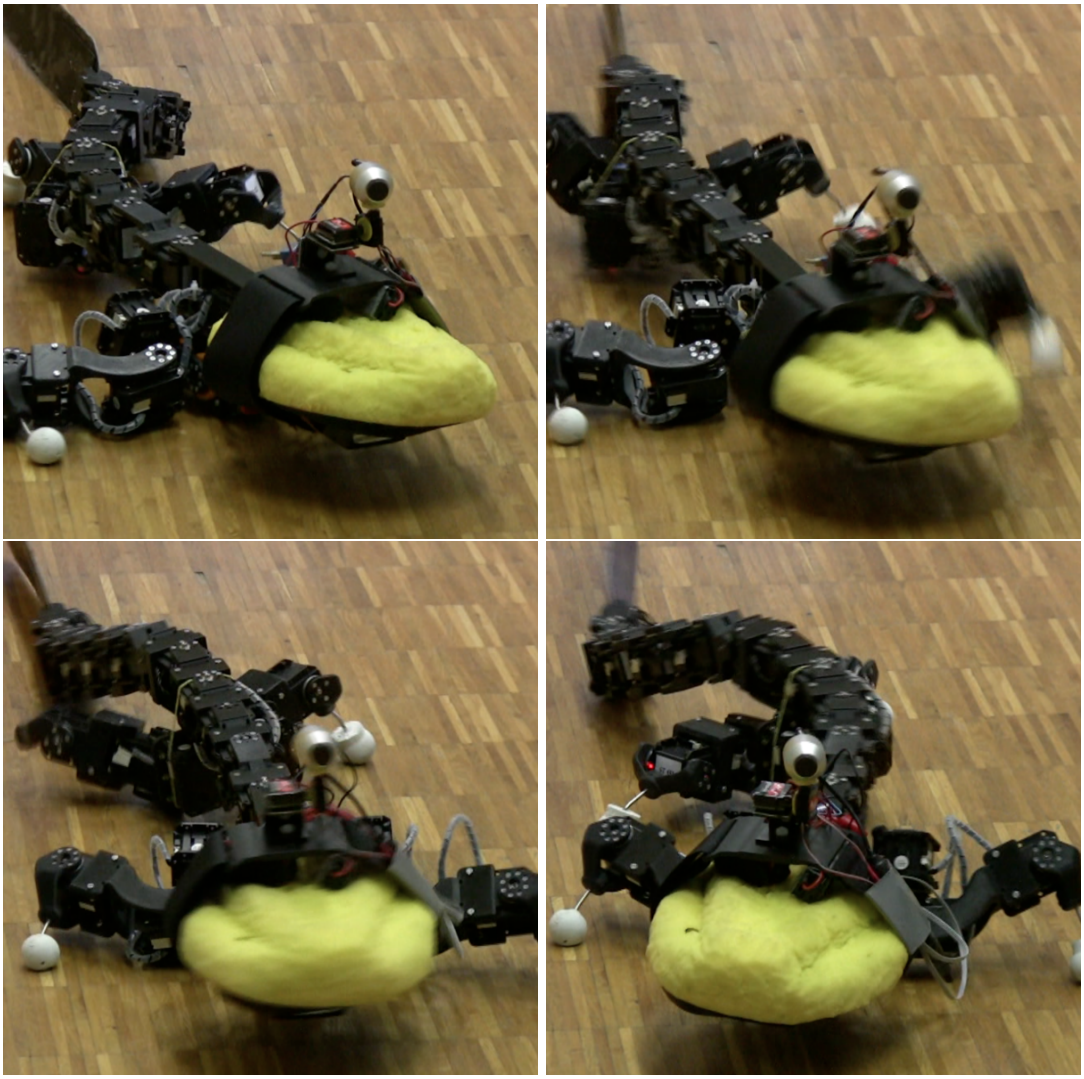


Figure 3.19: Snapshot of the Pleurobot walking with the stabilized camera attached to its head. Wile the head is moving a lot throughout one gait cycle, the camera is always pointing roughly in the direction of the motion (slightly to the right of the viewer).

Figure 3.20: Snapshot of the human subject experiment. Before convergence (up) the scene is moving a lot in the camera field of view. After convergence (bottom) the plant stays roughly steady, while the background seems to be rotating around it.

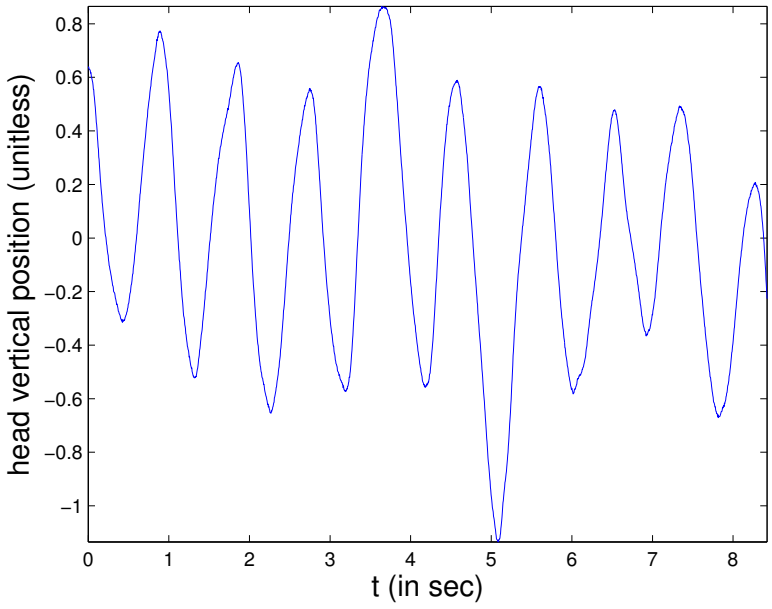


Figure 3.21: Vertical position of my head during running (unitless). This position was obtained by attaching a marker on my head and performing blob tracking.

into the dynamics of the designed oscillator such that changes in the parameters of the motion are tracked by the system. We showed that our system can be applied to stabilize the gaze when the robot is being moved, or when it is tracking a periodically moving object. We also showed that the system can successfully stabilize the head of the robot during actual locomotion, with a biped and an anguilliform robot, without the need for fast sensors. We demonstrated the performance of the system on a real robot tracking a periodically moving object. We developed a small wireless motorized camera to validate our approach on real robots and humans part of the physical *Robot Sensors* layer.

3.6.1 Discussion

The approach described in this chapter uses only vision as sensory feedback. This is a big advantage of the system since it can be applied to a wide variety of robots which do not necessarily have a large set of sensors. However, this is not a limitation of the system, and one could imagine fusing information from more sensors, depending on the application. A simple way to fuse information from a vestibular system and cameras, for instance, could be to use two different forcing signals for our oscillator. The forcing in Equation 3.8 would come from the vestibular system while the one in Equation 3.9 would come from visual cues. This would increase the speed and smoothness of the convergence of the frequency, while keeping the head motion phase locked with the optical flow.

Let us note that only the parameters of the nominal gait of the robot are learned. Unexpected fast changes of the head motion pattern are not stabilized. For some applications, this could be a downside of the system. However, during locomotion, fast changes of optical flow after head stabilization could be a sign that some unexpected events are occurring, e.g. the robot loosing balance, and this information could be used to trigger a response in the relevant control module.

Finally, the main limitation of the current system is the single shape of the output of the oscillator. We assume that the necessary compensatory motion to stabilize the head is close to a sine wave (note that the optical flow does not need to be sine wave, as long as the compensatory movements are). This is not always the case during locomotion. The shape of the corrective pattern is slightly modulated by the feedback term, but being able to generate the exact right oscillation patterns would surely increase the performance of our system.

3.6.2 Future work

While for this work we used mainly walking and swimming robots. It should be noted that this system would easily be applied to other periodically moving robots like

flapping flying robots.

Here we used optical flow, which is arguably the most general piece of visual information usable to do gaze stabilization. It does not rely on any specific object, color or complex visual processing. Yet, optical flow provides velocity information and as with all sensors providing velocity information, drift may occur. Alternatively, the position of an object or a blob could be used to force the AFO, which, although less general, would solve the drifting problem.

Future work should also include learning the whole shape of the robot motion. This could be done for instance by deducing the shape of the compensatory signals from the optical flow, and adding a dynamical filter to our oscillator (for instance of a combination of sine waves with different frequencies and amplitudes), or by designing an adaptive Gaussian mixture filter. Using a pool of coupled Adaptive Frequency Oscillators as done in [92] would be another solution to generate more complex shapes for the head motion.

We performed preliminary tests in simulation of our approach adapted to use a pool of oscillators instead of a single one for each degree of freedom. The experiment consisted in rotating the Hoap2 robot in the air as in Section 3.5.1 around its pitching axis. However, this time the motion applied to the robot was not a sine but a sum of three sines with frequencies 0.5Hz, 1.5Hz and 2.5Hz. The pool consisted of four oscillators initialized at frequencies 0.2Hz, 1Hz, 2Hz and 3Hz. The output of these oscillators was summed to generate the compensatory commands of the camera. Figure 3.22 and 3.23 show the motion applied to the robot, the evolution of the amplitude and frequency of each oscillator, the evolution of the optical flow and the compensatory commands generated for slow convergence parameters (small κ, β and η) and fast ones (high κ, β and η).

For slow convergence parameters, the oscillators nicely converge to the closed frequency from their initial one. The fact that two oscillators converge to the same frequency is not problematic at all since their amplitudes is simply going to be reduced. However, good stabilization is only achieved after $\approx 100s$, which is too much for our application.

For fast convergence parameters, the oscillators seem to converge faster, especially their amplitudes. However, before converging they seem to interfere with each other and start diverging to 0 amplitude. After sometime they start recovering and converge back. But no oscillator ends up converging to 0.5Hz, causing the stabilization to be imperfect. Several experiments have been performed with a different number of oscillators and different initial and final frequencies. But this sensitivity of the multi oscillator system with fast convergence seem to remain. Thus, as it is, this system is not usable for our application since with a real camera, additional sources of interference exist (noise, blur etc) which would cause the system to simply not converge.

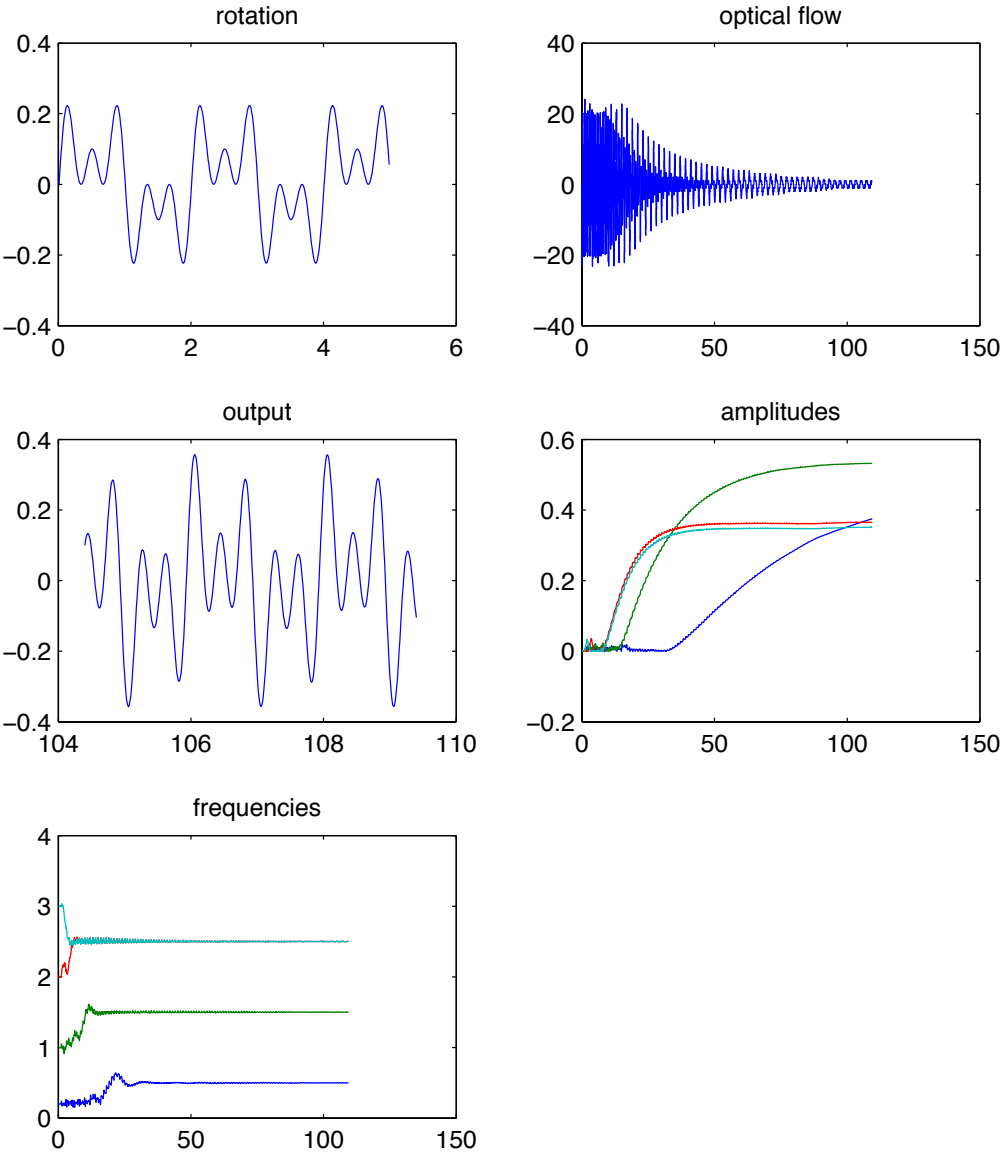


Figure 3.22: Multi-sine experiment with a pool of Adaptive Frequency Oscillators with slow convergence parameters (small κ, β and η). Rotation applied around the pitch axis of the robot (top left). Evolution of the optical flow (top right), the frequency (bottom left) and amplitude (center right) of the oscillators of the pool. Output generated by the pool (center left). The stabilization is successful but convergence takes a long time.

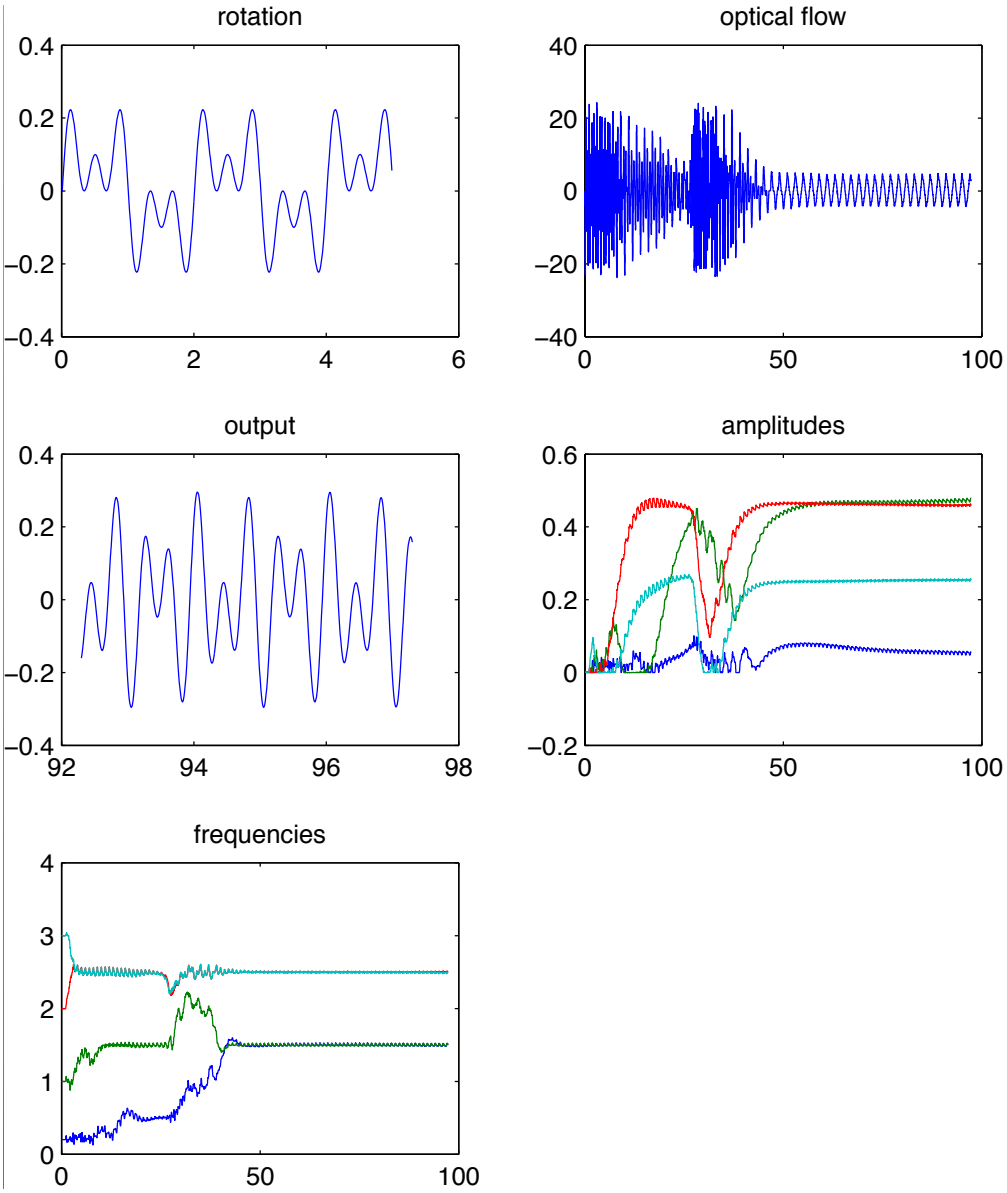


Figure 3.23: Multi-sine experiment with a pool of Adaptive Frequency Oscillators with fast convergence parameters (high κ, β and η). Rotation applied around the pitch axis of the robot (top left). Evolution of the optical flow (top right), the frequency (bottom left) and amplitude (center right) of the oscillators of the pool. Output generated by the pool (center left). The stabilization is only partially successful: the oscillators start converging to the right frequencies but then diverge and converge again. However after convergence, one frequency (0.5Hz) is not learned and the stabilization is not perfect.

Chapter 4

A Global Framework for Learning Robot Stability

This chapter serves as a technical introduction to the next two. It presents a global framework enabling a robot to learn its own stability. While this chapter considers the problem of stability in a global sense and describes only the general learning framework, the next two chapters will present concrete applications with specific robots and the derivations of this global framework to achieve specific tasks.

As in this chapter neither the robot nor its sensors are described in a specific way, the work presented here should be viewed as part of the *Mid Level Control* and *Low Level Control* (layers 4 and 5), as described in Chapter 1. Chapter 5 and 6 will give specific applications and thus deal with the *Sensor Processing* layer (layer 2) as well.

This chapter is a short one since it does not present results of the framework. Chapter 5 will present results of this learning framework applied to the control of posture of a compliant humanoid robot standing on a randomly moving platform, while Chapter 6 will show applications on a quadruped robot performing periodic locomotion on rough terrain.

This chapter starts addressing questions B, C and D described in Chapter 1. It describes what variables of the low-level control need to be modulated, how to interface vision with the lower layers, and how to fuse different sensors.

In this Chapter the central pattern generator controlling the robot is simplified to a standard Hopf oscillator in order to focus on the feedback mechanisms and learning. In the next two chapters, the CPG used will be a derivation of the Hopf oscillator.

4.1 Introduction

Stability is a central problem of locomotion control of legged robots. While their wheeled counterparts are assumed to be stable and hence usually have no mechanism to ensure stability, legged robots are meant to move on rougher terrains, with a typically higher center of mass to overcome obstacles, stairs, high slopes etc. While legged robots are supposed to deal better than wheeled robots with the difficulties of the terrain, they are also much more challenging to control. The higher position of the center of mass and the ability to position precisely their feet is a tremendous advantage over wheels to move around challenging environments, but are also the source of stability problems that have not been solved at this time. Whereas animals display impressive performance on very difficult terrains - think for instance of mountain goats - current legged robots are still far from the capabilities of even the poorest trained humans.

Stabilizing a robot walking on uneven ground or standing subject to external perturbations are two very similar problems in essence, which are usually tackled in a similar way. One very popular approach is to control the position of the so called *Zero Momentum Point* (ZMP) ([112]). Another method is to design reflexes aiming at minimizing the trunk angles of the robot. Arguably the most well known piece of work about reflex based stabilization is the one of Marc Raibert ([86]), which led to the Big Dog ([87]) stabilization mechanism. Another community aims at using reflexes on top of a Central Pattern Generator based controller ([57], [93]). These reflexes need to be implemented as mathematical equations in the CPG and are quite specific to a given task. A more thorough description of the state of the art of stabilization methods for legged robots can be found in Chapter 5 and 6. Designing reflexes in the CPG which are general enough to apply to a wide range of situations is a difficult problem which has only recently started being addressed. One option comes from Virtual Model Control ([85]), which attaches virtual springs at different points of the robot which stabilizing forces are projected through the Jacobian of the robot to the different actuated joints. Virtual Model Control has recently been implemented in the CPG of a quadruped robot with great results ([6]). VMC is however only based on a kinematic model of the robot and neglects dynamics effects.

In this thesis we take a slightly different approach. While we still rely on modulations of the CPG dynamics to stabilize the robot, we do not define these modulations a priori. Instead, we let the robot learn by trial and error the shape of these modulations of the CPG. By using an artificial neural network (ANN) as learnable mapping between the raw sensory information and the CPG modulations, and a reward-based learning procedure, we expect the following advantages over more conventional kinematics based approach:

- The system should be applicable to any robot with limited modifications (i.e. the number of oscillators in the CPG to match the number of degrees of freedom of the robot, the number of outputs of the ANN, the parameters of the gait). Only

the learning procedure should be redone.

- Since we let the robot free to learn ways of keeping balance, it should be able to find different strategies from the ones an engineer would implement.
- Our controller learns a direct non-linear mapping from sensors representing rates of change of the state of the robot (like gyroscope, accelerometer, optical flow etc.) and rates of change of the variables of the CPG. Thus with a proper learning process and a sufficient number of neurons, the learned neural network should approximate aspects of the dynamics of the robot (inertia, spring behaviors etc.), and not only its kinematics.
- This approach could also be used to control robots which are difficult to model accurately. This include robots with rigid links and non-linear compliant joints, but also robot with deformable structure, tensegrity robots etc.

4.2 Presentation of the system

In this section, we present in details the structure of our learning controller and the Central Pattern Generator. However, for the sake of clarity, the oscillator described here will be the standard Hopf oscillator, while we use a modification of this oscillator in Chapter 6 which allows to change the duty factor of the gait and apply a double peak trajectory to the knee joint. Please note that the Hopf oscillator was used in Chapter 2 for low-level control and that the Adaptive Frequency Oscillator used in 3 is also a derivation of this oscillator. Here we will not focus on the characteristics of this oscillator which have been well investigated in the literature, but we will insist on ways of including sensory feedback in its dynamics.

The equations of the Hopf oscillator are given below:

$$\dot{r} = \gamma(\mu - r^2)r \tag{4.1}$$
$$\dot{\phi} = \omega \tag{4.2}$$
$$x = r\cos(\phi) + o \tag{4.3}$$

where r is the radius of the oscillator, ϕ its phase, ω its frequency o its offset and x its output.

The output of the Hopf oscillator is a sine wave at convergence and is typically used as desired joint positions to control the robot. To achieve coordinated behaviors of the different joints of the robot, the oscillators are coupled by adding a term to Equation 4.2 as follows:

$$\dot{\phi}_{hi} = \omega + w_{ij}sin(\phi_j - \phi_i - \psi_{ij}) \tag{4.4}$$

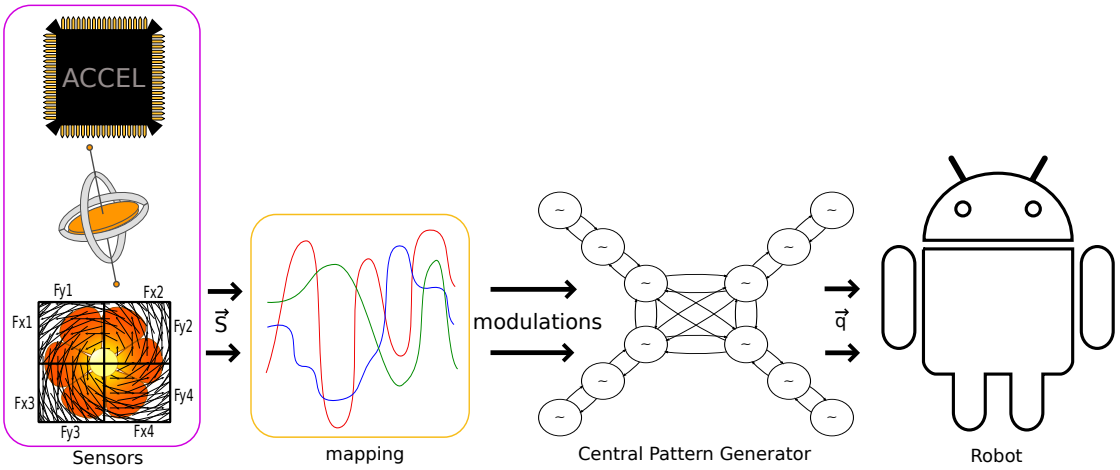


Figure 4.1: General idea of the problem for learning stability. Sensor values are processed from various sensors. The sensor values need then to be mapped to modulations of the CPG controlling the robot. The shape of the mapping from sensors to CPG modulations may be very non-linear and properly choosing this mapping is the main challenge here.

where ψ_{ij} is the desired phase difference between the oscillators controlling joints i and j and w_{ij} is a positive gain defining the coupling strength. This type of coupling is called diffusive as the term $\phi_j - \phi_i - \psi_{ij}$ disappears when the proper phase shift is reached and the coupling term cancels out.

4.2.1 Introducing sensory feedback in the Central Pattern Generator

The limit cycle of this oscillator is a perfect circle in phase space and its output after convergence is a perfect sine wave. Applying sine waves with proper phase shift to all joints of the robot allows for a rich variety of gaits but is not enough to cope with changing and uneven environments. Sensory feedback need to be included so as to morph this output. Figure 4.1 illustrates our vision of the problem of including sensory feedback in CPGs.

Sensors values are extracted from the robot sensors and need to be mapped to CPG modulations. Most methods found in the literature rely on a first step of state estimation relying on one or more sensors with heavy filtering and integration to extract a precise estimation of the state of the robot. A second step then consists of using this state to define modulations in the gait patterns or the posture of the robot to maintain balance. Both these steps constitute challenges in themselves and are usually high dimensional. In contrast we want to learn as direct a mapping as possible between sensors and actuators, effectively merging these two previously defined steps. The intuition behind is that the dimensionality of this direct mapping might be less than

the sum of the dimensionalities of the state estimator and controller.

Let us first investigate how to integrate sensory feedback in the oscillator described before. To modulate the shape of the output in a smooth way, we input sensory feedback only on the integrated variables of the oscillator. We hence make the offset an integrated variable as well to be able to influence each variable of the oscillator. Equations of the Hopf oscillator with sensory feedback are given next:

$$\dot{r} = \gamma(\mu + \kappa_r \mathcal{F}^r(\mathbf{S}) - r^2)r \quad (4.5)$$
$$\dot{\phi} = \omega + \kappa_\phi \mathcal{F}^\phi(\mathbf{S}) \quad (4.6)$$
$$\dot{o} = \kappa_o \mathcal{F}^o(\mathbf{S}) \quad (4.7)$$
$$x = r\cos(\phi) + o \quad (4.8)$$

where $\mathcal{F}^r(\mathbf{S})$, $\mathcal{F}^\phi(\mathbf{S})$ and $\mathcal{F}^o(\mathbf{S})$ are feedback functions of the sensor values $\mathbf{S}$.

The influence of these feedback functions on the respective variables of the CPG and its output are depicted in Figure 4.2.

The feedback on the radius $\mathcal{F}^r$ increases or decreases the amplitude of the oscillations temporarily. As soon as the feedback disappears the amplitude of the oscillations converges back to its default amplitude μ. This feedback can be used for instance to simulate leg retraction and extension reflexes.

The feedback on the offset $\mathcal{F}^o$ simply changes the setpoint of the oscillations, and stays encoded in the system when the feedback disappears. This feedback can be used for instance to change the posture of the robot when the slope of the ground changes.

The feedback on the phase $\mathcal{F}^\phi$ temporarily increases or decreases the oscillations frequency, by accelerating or decelerating the phase. When the feedback disappears, the apparent frequency of the oscillator goes back to its intrinsic frequency (the phase increases at the same speed as before the feedback arrived). This feedback can be used for instance to stop, or slow down temporarily the oscillations or to entrain the oscillator with an external signal. It is worth noting, as shown in the bottom right graph of Figure 4.2, that applying feedback to the phase of one hip influences the phase of the hips of the other legs. The amount of this influence is determined by the weight of the phase coupling w_{ij}.

Please note that by setting $r = 0$, the system can be used to generate purely discrete (i.e. non-rhythmic) movements, to perform postural adjustments in response to isolated perturbations (see Chapter 5). Otherwise, the system performs a combination of periodic and discrete movements, useful for instance to stabilize an open-loop gait on changing terrains (see Chapter 6).

Obviously simple boolean feedback functions as in previous figures are not sufficient to represent the richness of the necessary modulations of the CPG to cope with any environment. In the next section we explain how we chose to design these feedback

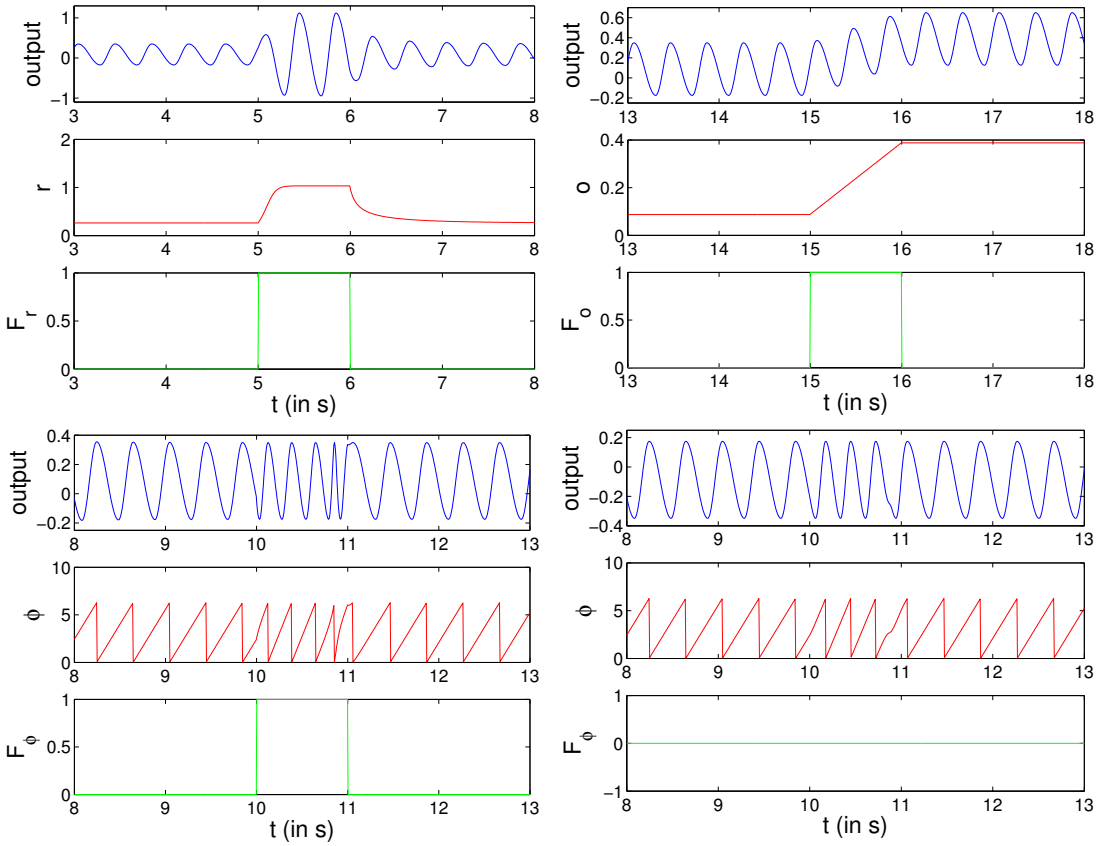


Figure 4.2: Influence of the feedback functions on the output. At $t = 5$ we set $\mathcal{F}^r = 1$ until $t = 6$, at $t = 10$ we set $\mathcal{F}^\phi = 1$ until $t = 11$, at $t = 15$ we set $\mathcal{F}^o = 1$ until $t = 16$. Top Left: influence of the feedback on the radius. Top Right: influence of the feedback on the offset. Bottom left: influence of the feedback on the phase. Bottom right: influence of the feedback on the phase of one oscillator on another coupled one, through the phase coupling.

functions.

4.2.2 Learning the feedback functions

Now that the low-level CPG has been defined the feedback functions $\mathcal{F}^r$, $\mathcal{F}^\phi$ and $\mathcal{F}^o$ need to be designed such that they map the sensor values to the right CPG modifications to stabilize the robot. One option is to design these feedback functions by hand, but this can be very complicated, especially when using multiple sensor values at once. Furthermore the feedback functions for the different legs might be strongly correlated. For instance, when standing on a platform, which starts tilting to the left, the robot might want to fold its right knees at the same time as it extends its left ones. Correlating these outputs is non trivial to do by hand. A linear mapping might not be sufficient with dynamic gaits on a compliant robot. For this work, we decided to represent these feedback functions with an artificial neural network (ANN), and particle swarm optimization to tune its weights.

The motivations behind are :

1. Neural networks can, with sufficient number of neurons, represent any mapping from an N-dimensional input vector to M-dimensional outputs.
2. Inputs and outputs of a neural network are, by essence, correlated and could thus reflect the correlation of the sensors and actuators.
3. A neural-network can learn a non-linear mapping from sensors representing velocities (gyroscope, optical flow etc.) to joint speeds (variables of the CPG), which should approximate aspects of the robot dynamics which a pure kinematic model cannot.
4. By learning, the robot might find different strategies to increase its stability from what engineers would imagine, or even from what animals would do.

Besides, a big advantage of CPGs for this work is that it decreases the control problem dimensionality to a small set of variables. This means that the number of outputs of our neural network, so the number of parameters to optimize to learn its weights is also limited. Thus we believe that CPGs are a good basis to learn a model-free mapping from sensory information to joint trajectories.

Figure 4.3 shows the full control framework including the learning process. First the controller extracts the sensor values from different sensors and performs very basic sensor processing (normalizing, low-pass filter). Theses sensor values are used as input to a fully connected perceptron with sigmoid activation, which outputs the values for $\mathcal{F}^r(\mathbf{S})$, $\mathcal{F}^\phi(\mathbf{S})$ and $\mathcal{F}^o(\mathbf{S})$ for each oscillator. The CPG takes these functions as sensory feedback and after integration outputs joint positions for each joint. The weights of the neural network W^D are tuned by a reward-based learning process, using particle swarm

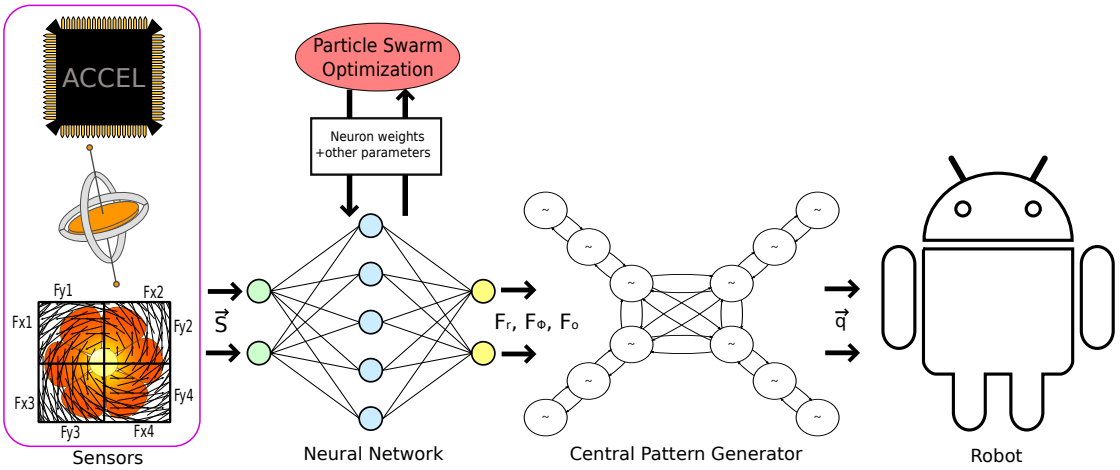


Figure 4.3: General idea of our framework for learning stability. Sensor values are processed from various sensors and fed to a fully connected neural network which weights are optimized using particle swarm optimization (PSO). This neural network outputs the feedback values for the CPG controlling the robot.

optimization (PSO). Other non-convex optimization algorithms like Genetic Algorithms or Simulated Annealing could of course also be used but we had good experience with PSO in rough fitness landscapes as considered here. Note that we cannot use supervised learning methods such as back-propagation here because we do not know the function to be learned (the different joint trajectories to achieve stabilization). We only know that the robot should maintain upright posture and not fall, but not how to achieve that. Together with the weights of the neural network, convergence parameters of the CPG and the slope of the sigmoid of the neurons are tuned since they also determine the influence of the feedback on the output trajectories. The total parameter vector for the optimization is: $[W^D, \gamma_r, w_{ij}, \kappa_r, \kappa_\phi, \kappa_o, \lambda]$, λ being the slope of the sigmoid $\frac{1}{1+e^{-\lambda x}}$ (typically we use the same values of the parameters $\gamma_r, w_{ij}, \kappa_r, \kappa_\phi$ and κ_o for all oscillators, and the same value of λ for all neurons).
For example, in Chapter 6, we use the Oncilla robot which has 12 controlled DOFs, 4 of them performing only discrete motions (the abduction/adduction joint of each leg). We thus need to generate 4 feedback functions ($\mathcal{F}^o(\mathbf{S})$) for the abduction joints and 24 feedback functions for the hip and knee joints ($\mathcal{F}^r(\mathbf{S})$, $\mathcal{F}^\phi(\mathbf{S})$ and $\mathcal{F}^o(\mathbf{S})$ for all 8 of them). Thus the neural network has 28 outputs. Assuming we use gyroscope as sensory input (i.e. 3 sensor values for the 3 rotational velocities), and a 10 neuron single layer neural network, the total number of weights for the fully connected neural network would be 310 ($3 \times 10 + 10 \times 28$), so $D = 310$. Adding the other parameters, the total number of parameters to optimize would be 316. When using vision as sensory input we typically have 8 sensor values (see Chapters 5 and 6), so $D = 360$ and the total number of parameters to optimize would be 366 ($8 \times 10 + 10 \times 28 + 6$).

A typical optimization scenario is then to initialize a first random population of these parameters, run the simulation with the robot performing the task considered, and record a measure of performance. The choice of fitness is crucial to ensure proper learning and will be addressed in the next section.

4.3 Choosing a fitness function

Choosing the right fitness function for the optimization problem is an important problem here. Since we typically have a few hundred parameters to optimize, a sub-optimal choice of fitness function can quickly misguide the search to wrong areas of the search space.

By experience we can derive the following properties for a good fitness function:

- Rather obviously, a good fitness function should be a good measure of performance of the robot for the considered task. For stability, we can consider several options: the inverse of the angles of the trunk, the inverse of the distance of the center of mass position to the zero position, or even the time before falling.
- A good fitness function should be monotonous. Indeed PSO tends to converge faster when two particles always have different fitnesses. For instance a boolean function of value 1 if the robot has fallen and 0 otherwise would not be a good candidate in our case, since many particles would have a fitness of 0 in the first iterations. Instead one should prefer the time before falling.
- The maximum value of the fitness function should be an ultimate goal but should not be achievable by the robot, so that it can keep improving throughout the optimization process.
- When using multiple criteria in the fitness function, the influence of each criterion on the value of the fitness should be carefully weighted. If one considers the example of a robot standing on a moving platform, if too much emphasis is put on the fact that the robot should not fall, and too few on the minimization of the trunk angles, the robot might choose a strategy such as kneeling, sitting or crouching on the platform without moving.

For the rest of this thesis, we used the following fitness function:

$$\mathcal{F} = \tau \left(\frac{1}{1 + \frac{1}{\tau} \int_{t=0}^{\tau} |\theta_P(t)| dt} \right)^{\xi_P} \left(\frac{1}{1 + \frac{1}{\tau} \int_{t=0}^{\tau} |\theta_R(t)| dt} \right)^{\xi_R} \tag{4.9}$$

where $\theta_P(t)$ and $\theta_R(t)$ are the absolute pitch and roll angle of the robot at time t, and τ is the time elapsed before the robot falls. ξ_P and ξ_R are constants used to give more or less importance to the minimization of the angles with respect to the time to fall τ.

While it would be possible to use only the time to fall as fitness function (see Chapter 4), we found that providing the trunk angles of the robot as performance helped in guiding the search and improved convergence. It is however not always desirable to minimize the trunk angle of the robot, for instance if it is walking down a slope. In such cases, the fitness function can be reduced to τ by simply setting $\xi_P = \xi_R = 0$. This fitness function is continuous, monotonous with respect to each performance measure τ, θ_P and θ_R, and the importance given to each performance criterion can be tuned with the parameters ξ_P and ξ_R. If the robot does not fall, the value of this fitness function monotonously tends to τ when θ_P and θ_R tend to 0. Please not that we do not try to perform actual multi-objective optimization (like optimizing a Pareto front for instance), but rather that we try to put more ore less emphasis on one part of the fitness function than others by tuning the parameters ξ_P and ξ_R. There is no guarantee that a certain constraint on one aspect of the fitness function (an upper bound for the trunk pitch angle for instance) is satisfied.

4.4 Conclusion

In this chapter we have presented our framework for learning stability in a general manner. This framework is in principle applicable to any robot with few modifications, capable of performing discrete postural adjustments or periodic gait modulations, and should be able to take into account some aspect of the dynamics of the robot even though the neural network itself is purely feedforward (no memory). In the next chapter we will investigate actual applications of this system to real robotics stability problems. We will describe the modifications that need to be done to this system to adapt it to different robots and different tasks. The next two chapters will also give insight into the role of fusing several sensors using the proposed system and how it affects its performance.

As mentioned in Chapter 1, central pattern generators are loosely inspired from the spinal cord of animals. While it is not a precise model, each oscillator approximates the behavior of a group of motor neurons in the spinal cord. They generate rhythmic patterns of locomotion sent to the "muscles" (the actuators) of the robot. To further this analogy, the neural network presented here may be viewed as a rough model of the motor cortex, getting processed sensor values from the visual and vestibular cortices (here the *Sensor Processing* layer) and sending descending signals to the spinal cord (the *Low Level Control* layer).

In this chapter we addressed question B described in Chapter 1 by described which variables of the CPG need to be perturbed in order to modulate the gait of the robot. The model we presented in this chapter is able to integrate different sensors and map them to low-level control values, hence addressing questions C and D.

In the following chapters, we use gyroscope and camera as sensory input. We also considered using accelerometer, which would have been more biologically inspired, but this would cause several issues. First, accelerometers include gravity, which would cause a constant offset to the input layer of the neural network. To enable the neural network to filter out this offset, its structure would have to be made more complex (at least two hidden layers). Second, the neural network outputs feedback on the variable rates of the CPG. This is analog to joint velocities which are then integrated to compute joint trajectories. As a pure static feedforward neural-network is unable to derive or integrate its inputs, it is easier to learn when using sensors providing also velocities as inputs. Therefore we chose to use the rotational rates provided by the gyroscope and the pixel velocities provided by the optical flow. Note that the neural-network would still most likely be able to learn using the accelerometer as input by doing a local approximation of its integral, but only preliminary tests have been performed so far.

While very general in its principle, this system has one main limitation: the need of a good physics-based simulator for the training phase (optimization). While simulators are available for most robots, and even-though we will call this method "model-free" in the remaining chapters, it should be noted that a simulated model of the robot is needed. However, only a *forward* model is needed and no *inverse* one (kinematic or dynamic).

In this thesis we present how this framework can be applied to learn robot stability. Note that for a small unpublished side project, we successfully applied this framework to a wheeled robot learning to navigate through a maze using optical flow.

Chapter 5

Vision for Postural Control

This chapter presents a derivation of the general learning framework described in Chapter 4 applied to the problem of stabilizing a humanoid robot standing on a randomly moving platform. The problem considered here is an example of learning discrete postural adjustments to compensate for external disturbances.

The work of this chapter is part of the *Sensor Processing*, *Mid Level Control* and *Low Level Control* layers (layers 2, 4 and 5 in Figure 1.3). As Chapter 4, it addresses questions B and C, i.e. what to modulate in the locomotion patterns in order to achieve the given task and how to interface vision with the lower layers. It also deals with aspects of sensor fusion (question D), here vestibular (gyroscope) and visual (camera) sensors. In this chapter we will use the Coman compliant humanoid robot, thus addressing question F.

While vision has a proven role in the human control of balance, as detailed in Section 5.1, it is rather underused for this problem in robotics, where vestibular sensors are usually preferred. The work of this chapter investigates whether vision can be used as efficiently as vestibular sensors to control the balance of a humanoid robot, or fused with them to further increase the performance.

The work presented here can also be viewed as a sub-part of the problem of stabilizing a gait. Indeed, the roughness of the ground (slope, uneven terrain) can be viewed as external perturbations applied to the robot, just like a moving platform. Whether walking or standing, the robot should perform postural adjustments to cope with these perturbations. However, since the robot is not walking here but just standing, only the offsets of the CPG will be modified to achieve the desired posture. In a sense we see the controller designed in this chapter as a walking controller with zero amplitude. The task here can be considered as easier as the on presented in Chapter 6, since when walking on rough terrain, the robot has to perform discrete postural adjustments, but also implement additional reflexes (on the CPG radius and phase) to deal with the inertia

caused by its locomotion patterns.

Most of the work presented in this Chapter was published in [41].

5.1 Introduction

In the last decades, vision, mostly optical flow, has been shown to play a role in unsuspected aspects of human motion like finely tuning the trajectory of the foot when stepping over an obstacle ([82]), estimating the traveled distance ([74]) and modulating walking speed and gait transition ([71]). For balance control, vision has even been shown to override the other sensory cues ([83], [66]). More specifically, optical flow has been identified as the main actor in the control of posture ([12]).

In robotics however, vision is generally highly under-used for the control of balance, other sensors like inertial measurement units (IMU) or force sensors being preferred. Most modern approaches for balance control rely on the Zero Momentum Point (ZMP) ([112]), or the Center of Pressure (COP), which are actually the same point viewed from two different perspectives ([44]). The ZMP is typically used to ensure stability by maintaining the center of gravity inside the support polygon. Alternatively, stability can be achieved by relying on the kinematic model. In [45] the authors designed kinematic synergies for the lower part of the Hoap2 robot which linearize the balancing control problem. Another kinematic based approach is Virtual Model Control (VMC) ([85]) which attaches virtual springs at different points of the robot and projects their generated forces through the Jacobian of the robot to the different joints. In [77], an optimization based method inspired form grasping is applied to balancing the DLR-KUKA humanoid robot. By distributing a force and torque among previously selected contact points on the feet, the robot is able to maintain a desired center of mass position.

To the best of our knowledge, approaches using vision for balance control are seldom. In [76] and [80], visual information is used to estimate the position of the ZMP and achieve robot stability. However both approaches rely on a reference object to estimate the pose of the robot.

In this chapter we present two approaches to the balance control problem of a compliant humanoid robot, the COMAN, standing on a moving platform. Figure 5.1 illustrates the problem we consider here.

The first approach presented here is a model-based kinematic approach using closed-form equations for each leg and ensuring that the feet do not slip while the trunk remains upright. The second one is a model-free optimization approach based on an artificial neural network mapping sensor values to joint velocities and integrators for the low-level control. In addition to the visual information, we also consider gyroscope values as sensory input in this chapter. Unlike the work mentioned before, the visual information here is not based on a particular object of the scene or special features like

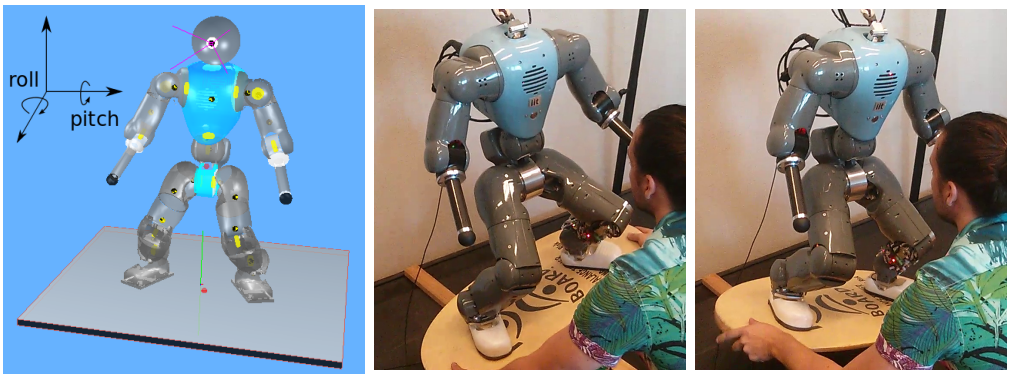


Figure 5.1: The problem tackled in this chapter: the compliant humanoid COMAN is standing on a moving platform and should maintain stable posture using either the gyroscope, the optical flow, or both.

the horizon orientation, but relies solely on the optical flow, which is very general and can be computed in nearly any environment. The goal of this chapter is also to compare the model-based and model-free approaches and investigate if vision can replace or be fused with a gyroscope to increase performance.

In Section 5.2 and 5.3, we present our control approach for model-based and model-free control of balance. Then, Section 5.4 explains how the camera images are processed to extract relevant information for each of the two approaches. Finally Section 5.5 presents experiments performed in Simulation and on the real COMAN robot.

5.2 Model-based control framework

The general control framework for the model-based postural control of the COMAN robot is depicted in Figure 5.2. COMAN is a 94.5 cm tall 25 DOF humanoid robot weighting 31.2kg. It features series elastic actuators in the shoulders, hips, knees and ankles (see Chapter 1, Section 1.5.2).

The pitching and rolling rotation speeds of the trunk of the robot are estimated using either the embedded gyroscope of the robot, or a camera placed on its neck. The process of computing the rotation angles using a camera is explained in Section 5.4. The rotation speeds are then filtered and integrated to extract an estimation of the absolute orientation of the platform (a simple Euler integration is used). This implicates that the initial posture of the robot has to be upright.

We call α_P and α_R the absolute pitch and roll angles of the platform and $\hat{\alpha}_P$ and $\hat{\alpha}_R$ their estimated value.

The absolute pitch and roll angles of the platform α_P and α_R fully determine the transformation matrices of the feet in the base reference frame of the robot such that

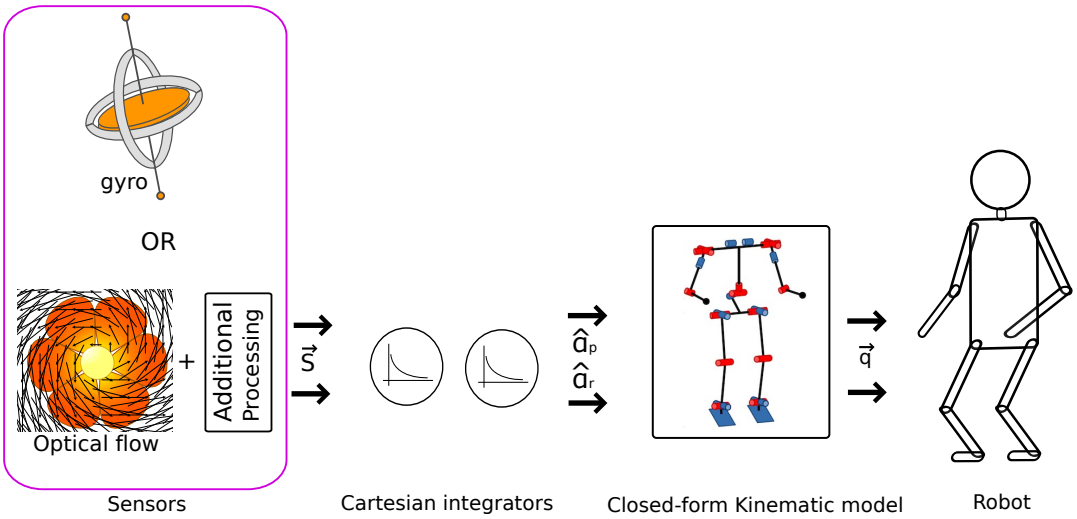


Figure 5.2: General control framework of the model-based approach. The absolute orientation of the platform is estimated from either the gyroscope values or the camera images. A closed form kinematic model controls the position of each joint so as to adapt the feet orientations and the leg lengths to keep the trunk upright and ensure that the feet do not slide on the platform.

the trunk is vertical. The transformation matrix T of the left and right foot is given below:

$$T = \begin{bmatrix} C_R C_y & -S_R & C_R S_y & p_x \\ S_P S_y + C_P C_y S_R & C_P C_R & C_P S_R S_y - C_y S_P & p_y \\ C_y S_P S_R - C_P S_y & C_R S_P & C_P C_y + S_P S_R S_y & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.1}$$

where $C_i = cos(\beta_i)$, $S_i = sin(\beta_i)$ and β_P, β_R and β_y are the pitch, roll and yaw angles of the feet being defined as $\beta_P = \hat{\alpha}_P$, $\beta_R = \hat{\alpha}_R$ and $\beta_y = \pm 25°$. This value of $25°$ is arbitrarily chosen to achieve a natural looking and stable posture, and its sign depends on the left or right foot. p_x, p_y and p_z are coordinates of the foot position in the root reference frame of the robot. The leg lengths should also be adapted to compensate for rolling motion of the platform, by folding the knees and adapting the other joint angles accordingly. These are computed so as to ensure that 1) the trunk keeps its initial orientation (upright, slightly bent forward) and 2) the feet do not slide on the ground. Geometrically we can deduce the target position $\mathbf{p} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$ for the left

and right feet according to the initial posture and the estimated platform rotation:

$$\mathbf{p} = \begin{bmatrix} p_x^0 \cos\hat{\alpha}_R \pm L_W(\cos\hat{\alpha}_R - 1) \\ p_y^0 + (p_x^0 \pm L_W)\sin\hat{\alpha}_R \\ p_z^0 \end{bmatrix} \tag{5.2}$$

where $\mathbf{p^0} = \begin{bmatrix} p_x^0 \\ p_y^0 \\ p_z^0 \end{bmatrix}$ is the initial position of the foot in the hip root reference frame when the platform is not rotated and L_W is half the waist width of the robot. The signs are chosen accordingly for the left and right foot.

Once the target positions of the feet are chosen, various inverse kinematics methods can be used to achieve them. The most common ones rely on Jacobians, or use non-linear numerical solvers. This however introduces delays in the computation and can be computationally expensive. However since each leg is a 6-DOF manipulator and we now fixed the position and orientation of the end-effector (the foot), closed form equations can be derived. Various methods exist to derive these equations usually involving computing the symbolic inverse transformation matrix, and squaring, adding and dividing some of its components to decouple the degrees of freedoms as much as possible. This can be a difficult and time consuming process and papers have been published on the sole matter of deriving closed form equations for the inverse kinematics of humanoid robots ([15], [8]). Furthermore, the order of the joints of the COMAN robot (hip pitch, hip roll, hip yaw, knee, ankle roll, ankle pitch) is not standard for humanoid robots and makes it difficult to apply these methods. Fortunately, a library named IKFast, part of the open source framework OpenRave, has been developed which is capable of computing the closed form equations for any 6-DOF manipulator and for special cases up to 8-DOF. It takes care of all the singularities and, due to redundancies, outputs up to six solutions for the inverse kinematics. These solutions are then narrowed down to one by taking into account the joint limits.

5.3 Model-free control framework

The general control framework for the model-free postural control of the COMAN robot is depicted in Figure 5.3. This framework is a derivation of the system previously presented in Chapter 4. It is the discrete movement version, equivalent to having the amplitudes of each oscillator set to 0 and using only their offset. Thus the components controlling the joint positions here are called *integrators* instead of *oscillators* since they do not effectively oscillate but output discrete trajectories in a smooth way. Their equations can be reduced to :

$$\dot{o} = \kappa_o \mathcal{F}_o(\mathbf{S}) \tag{5.3}$$

Here o directly determines the output (the joint position of the oscillator).

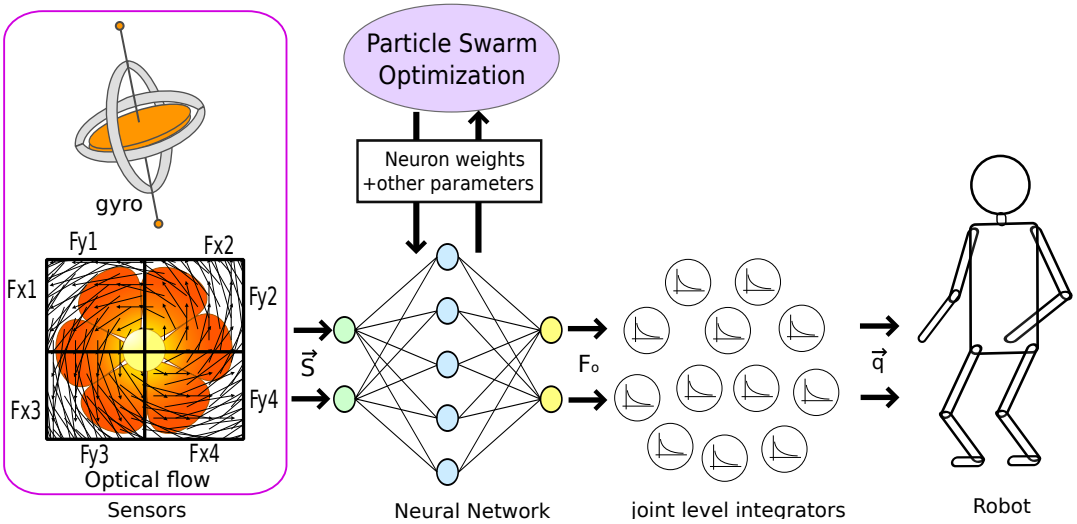


Figure 5.3: General control framework of the model-free approach. The sensory information composed from down-sampled optical flow and/or gyroscope values is fed to a neural network outputting feedback to integrators for each joint. The weights of the neural network are optimized using particle swarm optimization.

The sensory information here consists of the gyroscope values and a down-sampling of the optical flow computed as described in Section 5.4. As in the model-based approach (Section 5.4), we could use the estimates of the rotational velocities as input to the neural network. However, neural networks can deal with less explicit sensory input like the raw optical flow, which would save one sensor processing step. Furthermore the neural network may also extract other relevant information from the flow field. By the nature of our system, gyro and camera can be used separately or fused seamlessly by providing them both as input to the neural network. Since only rotation velocity information is used, as for the model-based approach, the controller has to be started with the robot upright.

As explained in Chapter 4, these sensor values are used as input to a fully connected neural network with sigmoid activation, which outputs the values $\mathcal{F}_o(\mathbf{S})$ for each joint. Hence this neural network can be viewed as a learnable non-linear mapping between sensor values and joint velocities. Here, we chose to control 6 DOF for each leg and 2 DOF (shoulder hip and roll) for each arm, so 16 DOF in total. The sensor vector consists of 3 values for the gyroscope and typically 8 values for the down sampled optical flow (x and y components for each quarter of the image). The integrators for the joints take these functions as sensory feedback and output joint positions for each joint. The weights of the neural network W^D are tuned by a reward-based learning process, using particle swarm optimization (PSO). Together with the weights of the neural network, the convergence rate of the integrators and the slope of the sigmoid of the neurons are tuned since they also determine the influence of

the feedback on the output trajectories. The total parameter vector for the optimization is: $[W^D, \kappa_o, \lambda]$, λ being the slope of the neurons sigmoid activation function $\frac{1}{1+e^{-\lambda x}}$

We use the fitness function described in Chapter 4 with $\xi_P = \xi_R = \xi = 5$, to give the same importance to the minimization of the pitch and roll angles, with respect to the time to fall:

$$\mathcal{F} = \tau \left(\frac{1}{1 + \frac{1}{\tau} \int_{t=0}^{\tau} |\theta_P(t)| dt} \right)^{\xi} \left(\frac{1}{1 + \frac{1}{\tau} \int_{t=0}^{\tau} |\theta_R(t)| dt} \right)^{\xi} \tag{5.4}$$

where $\theta_P(t)$ and $\theta_R(t)$ are the absolute pitch and roll angle of the robot at time t, and τ is the time elapsed before the robot falls. ξ is a constant used to give more or less importance to the minimization of the angles with respect to the time to fall τ.

5.4 visual-processing

Images from the camera are processed so as to obtain optical flow information. We use the Lucas-Kanade ([67]) implementation in OpenCV to compute the optical flow. This is a very high-dimensional information (typically around 500 vectors here), and needs to be reduced. For the model-free approach, we want to investigate if the neural network can process non-explicit information like the angular rates of the gyro or the raw optical flow. Thus we just need to down-sample the optical flow to tractable values in term of number of neural network input. Here we choose to split the image into two by two quarters and compute the averaged flow in each of them. This leads to four vectors having two components each, so eight values.

For the model-based approach however, this is not enough since we need to have an accurate estimation of the absolute platform rotation. Here the optical flow can be decomposed into the part produced by the pitching motion and the one caused by the rolling motion of the camera. Since the robot is standing on a platform only subject to rotations, the translational part of the optical flow is neglected here. Figure 5.4 shows what these components look like.

As shown in [21], each vector of the optical flow V satisfies the following equations:

$$V = V_R + V_P = \begin{bmatrix} 0 \\ k\omega_P \end{bmatrix} + \begin{bmatrix} -\omega_R(y_P - y_c) \\ \omega_R(x_P - x_c) \end{bmatrix} \tag{5.5}$$

where ω_P and ω_R are the pitching and rolling rotation speeds, and k is a constant depending of the characteristics of the camera. $p \begin{bmatrix} x_P \\ y_P \end{bmatrix}$ is the origin of the considered vector in the image coordinates, and $c \begin{bmatrix} x_c \\ y_c \end{bmatrix}$ the center of the rolling rotation.

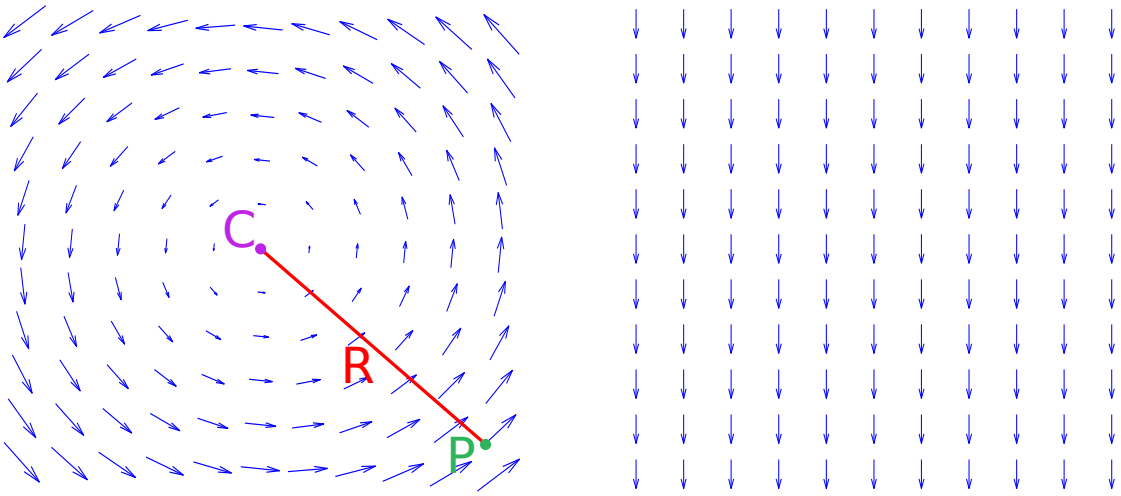


Figure 5.4: A representation of the components of the optical flow caused by rolling (left) and pitching (right) rotations.

Subtracting vector flows pairwise we can decouple the rolling component of the flow, since the pitching component is constant over the whole image.

$$\Delta V = \begin{bmatrix} -\omega_R \Delta y \\ \omega_R \Delta x \end{bmatrix} \tag{5.6}$$

Each pair of vector satisfying this equality, we can apply a simple mean square error regression to estimate ω_R, and successively get ω_P.

5.5 Results in simulation

In this section we present results using the model-based and the model-free approach in simulation. Preliminary results using the real robot are given in Section 5.6. We use the Webots simulation environment ([116]) with a very realistic model of the COMAN robot. This model simulates the series elastic actuators and whole body measurements on the robot and the simulator displayed very similar behavior. The real COMAN robot does not have a camera or even a head to this date. We therefore added a massless head and camera to the model. To match those achievable on the real robot, the time step of the controller and the gyro data was set to 8ms (125Hz), and that of the camera to 24ms (∼40Hz).

For both approaches the testing environment was the same. We placed the robot on

a platform rotated according to the following rules:

$$\alpha_P = A\sin(2\pi F_P t) \quad (5.7)$$
$$\alpha_R = A\sin(2\pi F_R t) \quad (5.8)$$
$$A = \eta t \quad (5.9)$$

where F_P and F_R are the pitching and rolling frequency, and A the amplitude of the oscillations. The amplitude gradually increases with time so that the stabilization problem becomes harder. Here we set $\eta = 0.007$ so that at $t = 20s$ $A \approx 8^\circ$ (total amplitude of 16 degrees).

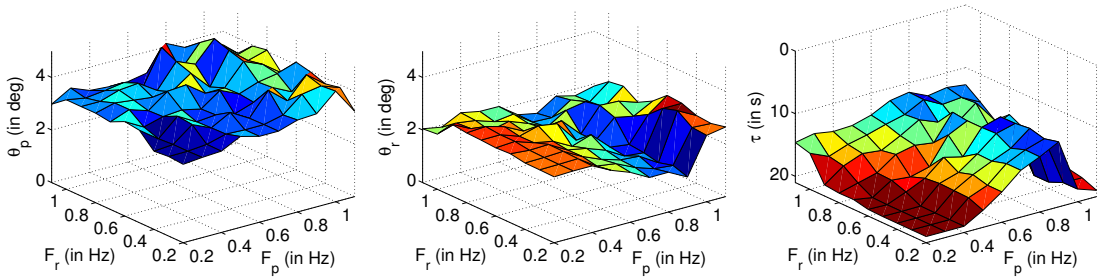
5.5.1 Model-based approach

We ran simulations with every combinations of pitching and rolling frequencies ranging from 0.2Hz to 1.1Hz with steps of 0.1Hz (100 tests in total). For each simulation we recorded the trunk pitch and roll angles and the time to fall. The results are presented in Figure 5.5. Small angles mean that the trunk is upright, so that the stabilization is more successful. Large time to fall values mean that the controller is able to cope with larger movements of the platform, since their amplitude is gradually increasing with time. Since the total simulation time is 20s, a time to fall of 20 is considered a success (the robot did not fall). When not controlling the robot, the success rate is 14%. This corresponds to low frequencies, where the initial posture is stable enough to cope with the movements of the platform. The success rate of the model-based controller using gyro as input over all the trials was 60%, while the one using vision as input reached 71%. However, the controller using gyro input was more successful than the one using optical flow at reducing the rolling and pitching motion with an averaged pitch angle of 1.47 degrees against 1.69 degrees and an averaged rolling motion of 0.27 degrees against 0.98 degrees.

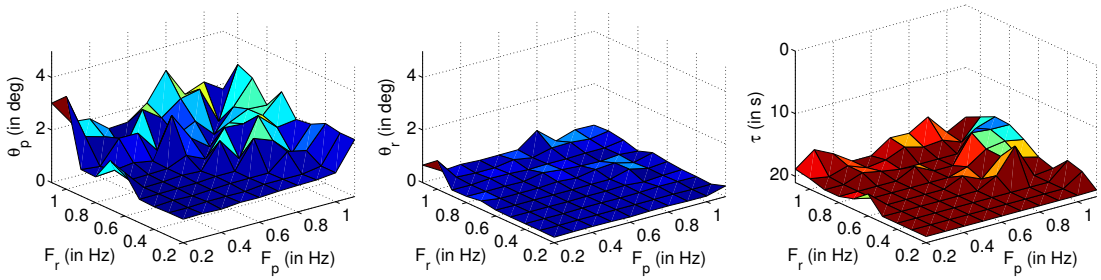
5.5.2 Model-free approach

The model-free approach requires a phase of training. This phase consisted in running a PSO algorithm with 100 particles for 300 iterations maximum. Each particle is a set of parameters (weights, sigmoid slope and convergence rate) defining the neural controller. We performed experiments in simulation and tried different training scenarios. For the first scenario, we trained the controller with a platform pitching frequency of 0.7Hz and a rolling frequency of 1Hz. For the second one, the controller was trained simultaneously with three combinations of pitching and rolling frequencies : 0.3Hz - 0.7Hz, 0.7Hz - 0.5Hz and 1Hz - 1Hz. This means each PSO particle was repeated three times and its fitness was computed as the average of the fitness for each frequency combinations. The last scenario consisted in training the controller with six combinations of frequencies: 0.3Hz - 0.7Hz, 0.7Hz - 0.5Hz, 1Hz - 1Hz, 1Hz - 0.3Hz, 0.5Hz - 0.3Hz and 0.7Hz - 1Hz.

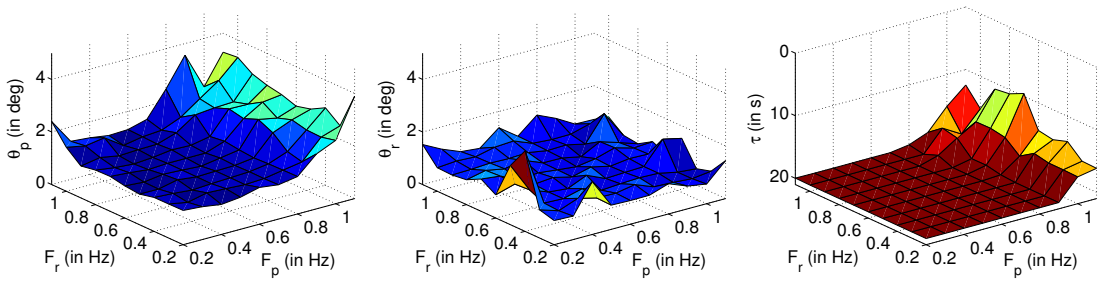
Each scenario was repeated using either only the gyro as sensory input, or the camera, or camera and gyro together. To make sure results do not depend on initial



(a) Robot not actuated



(b) Model-based controller with gyroscope input



(c) Model-based controller with camera input

Figure 5.5: Average trunk pitch (left), average trunk roll (middle) and time to fall (right, z axis inverted) without control (a), with the model-based controller using gyro values (b) and with the model-based approach using camera (c). All tests were performed in simulation and lasted 20 seconds.

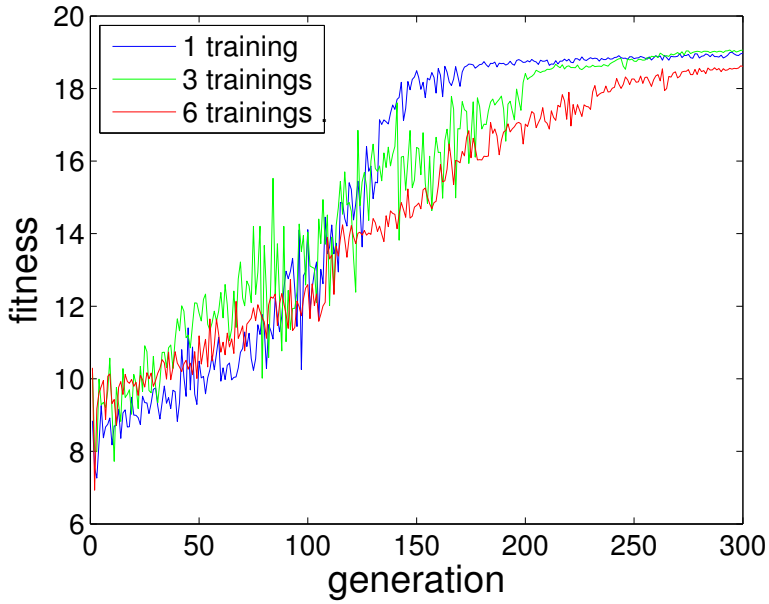


Figure 5.6: Evolution of the fitness for each of the 3 scenarios: training once, training 3 times and training 6 times

conditions of the PSO algorithm, we repeated each scenario three times, and got qualitatively similar results. The evolution of the maximum fitness function for each scenario using the gyro as sensory feedback is shown in Figure 5.6. The optimizations take between 150 and 300 iterations to converge but the more repetitions the training has, the longer the optimization takes to converge. The same property can be observed with camera or camera and gyro as sensory input.

Finally we ran generalization tests for each of the scenarios. As for the model-based approach, we used frequencies ranging from 0.2Hz to 1.1Hz with steps of 0.1Hz, for roll and pitch (100 tests in total). The results are shown in Figure 5.7. We define success as $\tau = 20s$, meaning the robot did not fall during the whole simulation. When not controlling the robot, the success rate is 14%, as mentioned in Section 5.5.1. When the robot is controlled with the neural network trained with only one pair of frequencies, the performance climbs to 73%. When increasing the number training frequencies to three the success rate actually decreases to 61%. This decrease of performance between the first and second scenario is surprising, since in principle increasing the number of training sample should increase the generalization score (for a very high number of training samples, over-fitting problems might arise but this is clearly not the case here with only three pairs of frequencies). Actually, the generalization performance does not decrease, as defined by the fitness function presented in Section 5.3. Indeed, while the averaged time to fall is indeed lower for the second scenario (19.02 vs 19.34 for the first one), the averaged rolling and pitching angles are also lower (0.56 vs 0.64 degrees for the pitch and

0.25 vs 0.38 degrees for the roll), making the overall fitness of the second scenario higher. When using six pairs of frequencies for training, the performance climbs again to 90%. The controller is able to generalize to most unseen cases, even slightly outside the training bounds (frequencies of 0.2 and 1.1Hz). Furthermore, except where both the pitching and the rolling frequency were above 0.9Hz, nearly all (97%) of the trials were successful.

Figure 5.8 shows the same performance indicators as before for the model-free method using different sensory inputs : gyro alone, vision alone and gyro and vision together. The controller was trained with three different frequencies. Here as for the model-free approach the performance using the camera was slightly higher than the one using the gyro (64% against 61%). This is interesting considering the visual input is simply the down-sampled raw optical flow. When combining gyro and camera together, the success rate raised to 75%. The minimization of the angles was not as good when using the camera as when using the gyro only. One explanation for this might be the greater latency of the camera sensor.

5.5.3 Adaptability to discrete movements

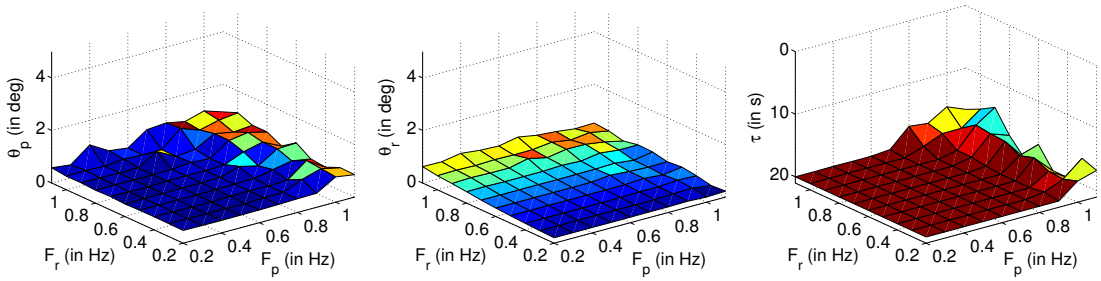
In this section we investigate how our model-free approach can generalize to a different kind of movements. While the training and generalization investigations performed in the last section were dealing with periodic movements (the platform trajectory followed a sine wave), we want to verify that our system can compensate for discrete movements as well. These discrete movements tests are necessary to make sure that the model learned by our system is not dependent on the fact that the robot is constantly moving, and that it can cope with a still robot and isolated movements.

This experiment is similar to the last one in the fact that the robot is standing on the same platform in the same environment. We used here the result of the last scenario of Section 5.5.2, i.e. gyro used as sensory input and trained with six different frequencies. No relearning or tuning of any kind was done here. The controller was left untouched. Results with other scenarios and sensory inputs should be qualitatively similar, since the system structure and learning procedure was the same. The movements applied on the platform were evenly spaced in time and consisted in discrete rotations in pitch and roll, with random directions, and fixed amplitude. These rotations followed a trajectory defined by the following equations:

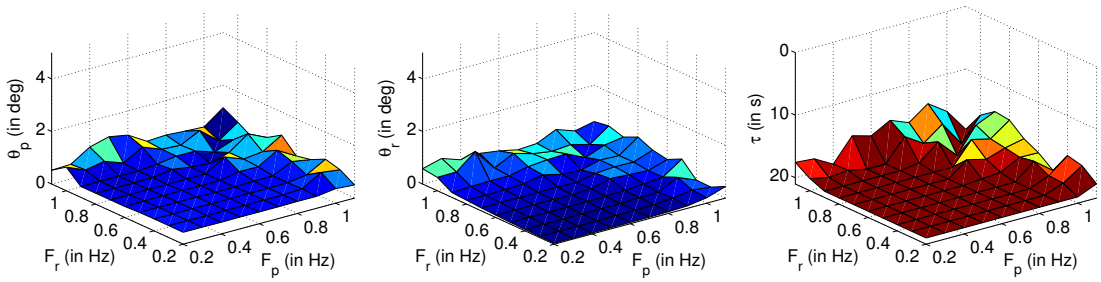
$$\alpha_P = a_P \tanh(2\pi\nu(t - \frac{\tau_d}{2})), t = [0, \tau_d] \tag{5.10}$$

$$\alpha_R = a_R \tanh(2\pi\nu(t - \frac{\tau_d}{2})), t = [0, \tau_d] \tag{5.11}$$

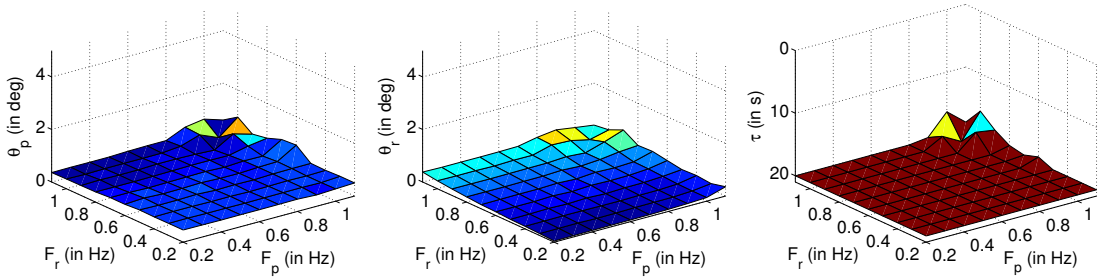
where a_P and a_R are the pitch and roll amplitudes of the movement, chosen randomly, and normalized so that $A_P/2 < a_P < A_P$ and $A_P/2 < a_P < A_P$. A_P and A_R are fixed over one test, and varied across all tests. These hyperbolic tangent trajectories



(a) Controller trained with one pair of frequencies (pitch: 0.7Hz roll:1Hz)

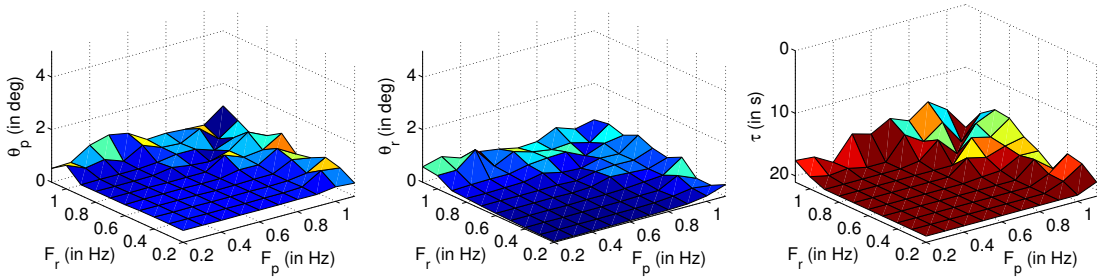


(b) Controller trained with three pairs of frequencies (pitch/roll: 0.3/0.7Hz - 0.7/0.5Hz and 1/1Hz)

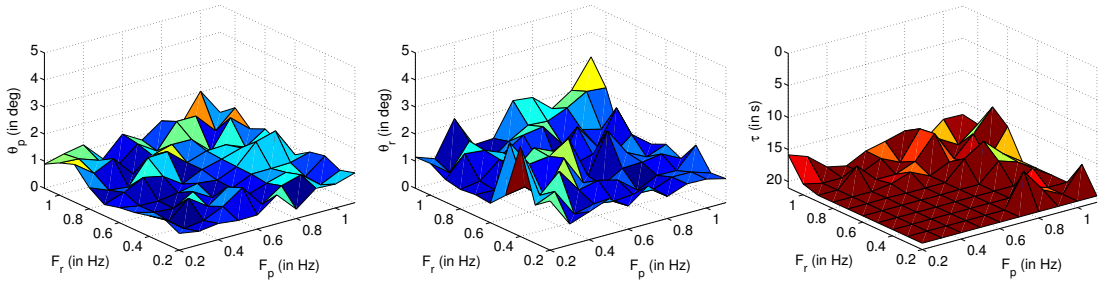


(c) Controller trained with six pairs of frequencies (pitch/roll: 0.3/0.7Hz - 0.7/0.5Hz - 1/1Hz - 0.3/0.5Hz - 0.3/0.7Hz and 0.7/1Hz)

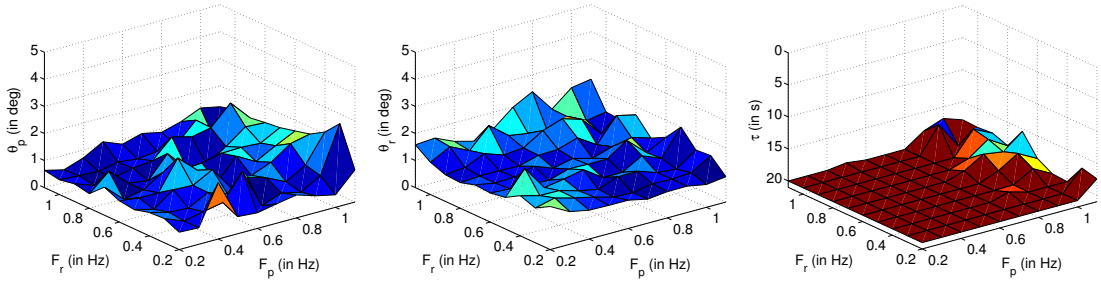
Figure 5.7: Average trunk pitch (left), average trunk roll (middle) and time to fall (right, z axis inverted) for each of the three scenarios (a, b and c). All generalization tests were performed in simulation and lasted 20 seconds.



(a) Gyroscope input only



(b) Optical flow input only



(c) Gyroscope and optical flow fused input

Figure 5.8: Average trunk pitch (left), average trunk roll (middle) and time to fall (right, z axis inverted) for the model-free approach using gyro only (a), optical flow only (b) and gyro fused with optical flow (c). All generalization tests were performed in simulation and lasted 20 seconds.

ensure a bell shaped velocity profile. ν is the slope of the hyperbolic tangent function which effectively determines the speed of the movement (i.e. higher values of ν mean higher motion speed and thus lower durations of movement). τ_d is the minimum time between two movements, here fixed to 3 seconds. The actual time between movements also depends of ν. We investigate here the performance of our controller against all combinations of A_P and A_R varying from 0.05 to 0.25 radians (about 3 to 14 degrees). We consider three different cases: movements with slow speeds ($\nu = 0.3$), medium speeds ($\nu = 0.7$) and high speeds ($\nu = 1.2$). Figure 5.9 shows examples of the trajectories applied to the platform for each cases. For fast speed, the movements are clearly isolated and the platform is still in between, while for slow speeds the platform is almost constantly moving.

We performed systematic tests in simulation to measure the performance of our controller when coping with these discrete platform movements. The amplitude of the rolling and pitching motion of the platform were varied from 0.05 to 0.25 radians across all the tests with steps of 0.02 (100 tests total). We used the same performance measures as in the previous sections, namely the averaged trunk pitch and roll angles, and the time to fall.

Figure 5.10 shows these performance measurements for slow, medium and fast movements, when no control is applied, for reference. Figure 5.11 shows the same performance measurements with our model-free learned controller. For all cases, the closed-loop controller greatly improved the performance over not controlling the robot. The trunk angles were greatly decreased and the robot did not fall for almost all amplitudes when the motion of the platform was slow or medium. For fast movements, the robot could cope well with low amplitudes, but not with higher ones (> 0.15 radians, > 9 degrees). As for periodic platform movements, we notice a higher sensitivity of the controller to high pitching amplitudes than high rolling amplitudes. This can be explained by the bigger size of the support polygon in the rolling direction than the pitching one, due to the distance between the feet.

5.6 Real robot experiments

5.6.1 Model-based

We started implementing the model-based approach on the real robot. For now we only considered the rolling angle, since it is less subject to spring excitation by the control. We recorded the rolling angles over a 25s experiment with and without the controller. The platform was moved by hand in a close to rhythmic fashion. Results are shown in Figure 5.13. Please note that the motion of the platform was done by hand, and although we tried to apply the same amplitude of motion with and without the control, these motions obviously differ. Furthermore we had to make the speed of the motion without control very low in order to prevent the robot from falling. The amplitude of the rolling motion with the control was reduced by more the 50% compared to not

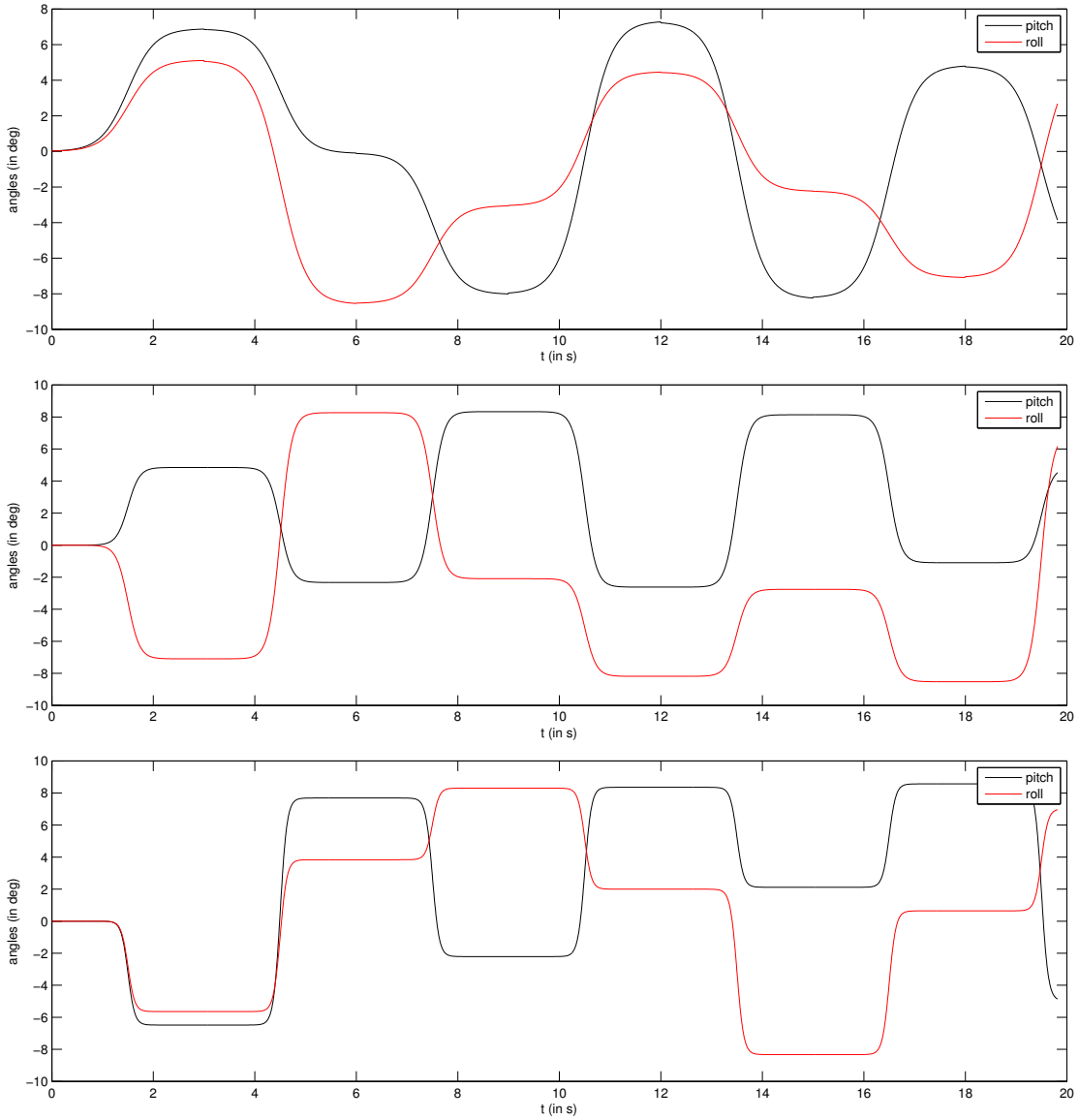
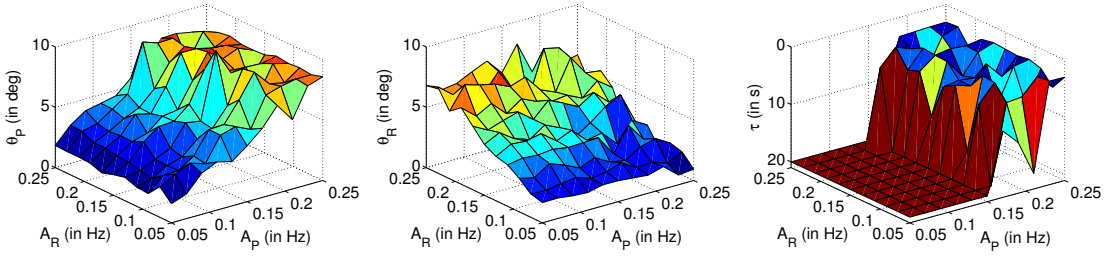
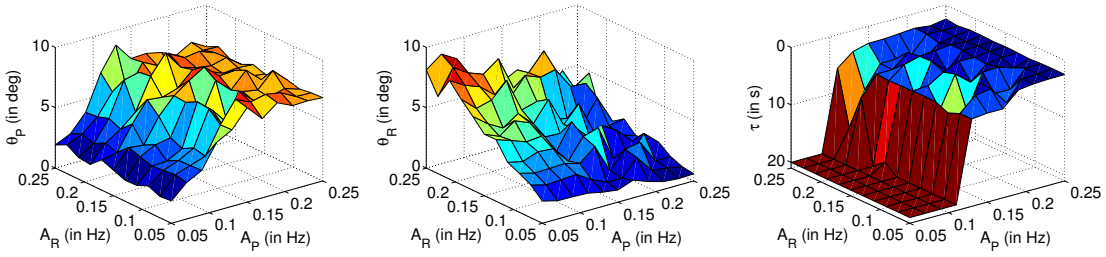


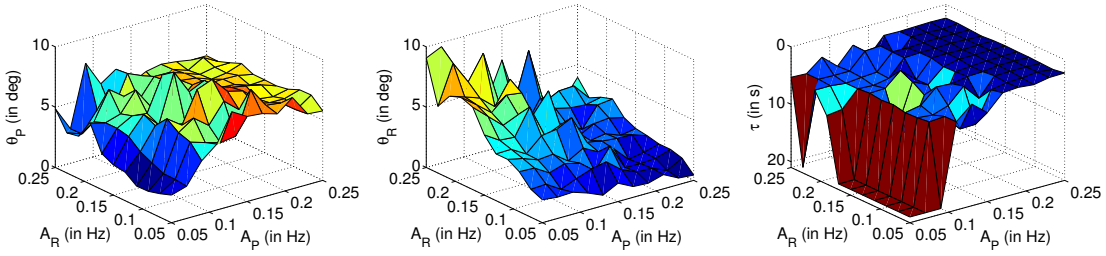
Figure 5.9: Example of trajectories applied to the pitch and roll angles of the platform for slow speed motions (top, $\nu = 0.3$), medium speed (middle, $\nu = 0.7$) and fast speed(bottom, $\nu = 1.2$). The amplitude of the motion is random and constrained in the intervals $[\frac{A_P}{2}; A_P]$ and $[\frac{A_R}{2}; A_R]$. The platform movements are evenly spaced in time 3 seconds apart.



(a) Slow platform movements ($\nu = 0.3$)

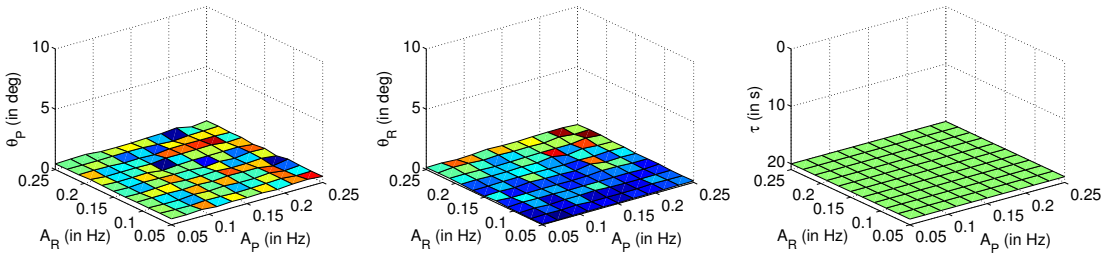


(b) Medium speed platform movements ($\nu = 0.7$)

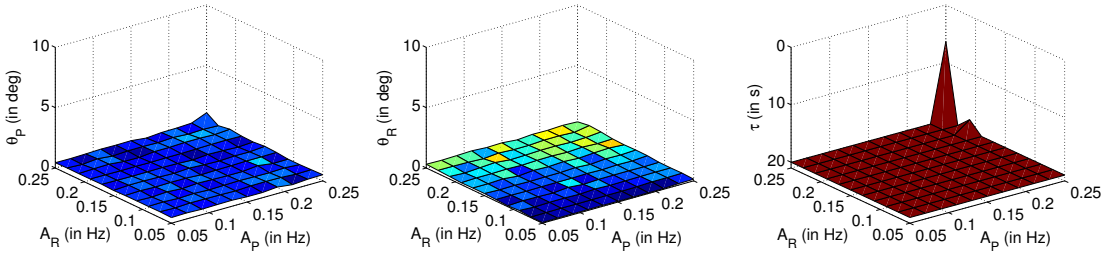


(c) Fast platform movements ($\nu = 1.2$)

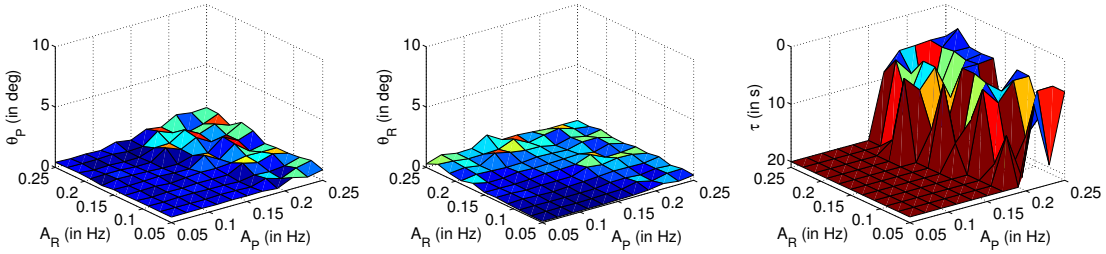
Figure 5.10: Results for discrete platform movements with varying rolling and pitching amplitudes A_P and A_R, when no control is applied. Average trunk pitch (left), average trunk roll (middle) and time to fall (right, z axis inverted) for different platform movement speeds. All tests were performed in simulation and lasted 20 seconds.



(a) Slow platform movements ($\nu = 0.3$)



(b) Medium speed platform movements ($\nu = 0.7$)



(c) Fast platform movements ($\nu = 1.2$)

Figure 5.11: Results for discrete platform movements with varying rolling and pitching amplitudes A_P and A_R, with the model-free approach with gyro input. Average trunk pitch (left), average trunk roll (middle) and time to fall (right, z axis inverted) for different platform movement speeds. All tests were performed in simulation and lasted 20 seconds.

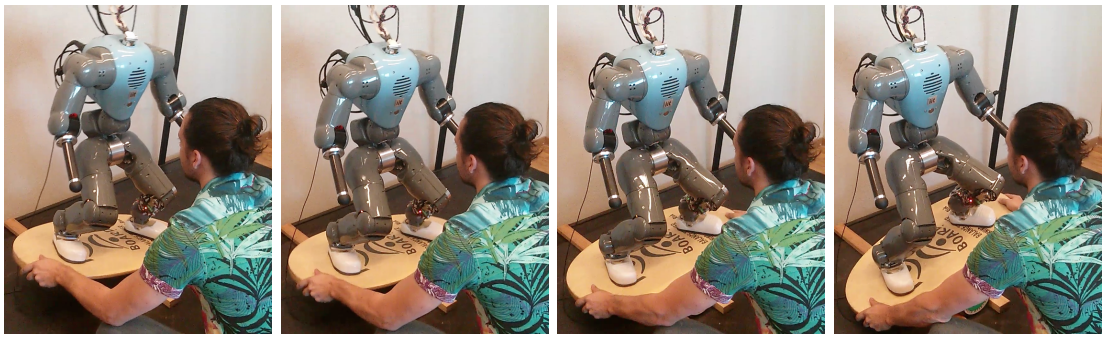


Figure 5.12: Snapshots of the model-based approach applied on the real Coman robot. When the platforms moves the trunk of the robot stays roughly upright.

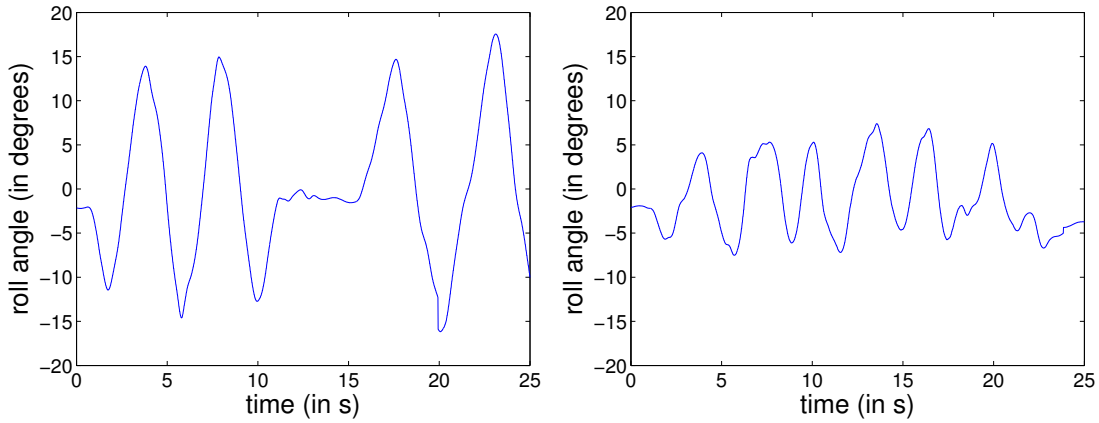


Figure 5.13: Rolling angle of the real COMAN robot for a 25s experiment without control (left) and with the model-based approach (right). The motion of the platform is applied by hand, so both motions differ, but the amplitudes of the platform movement should be roughly the same.

controlling the robot. The video at [2] shows the model-based approach applied on the real robot, and snapshots of it can be seen in Figure 5.12.

5.6.2 Model-free

The model-free approach was much easier to port on the real robot than the model-based one. The controller using gyroscope as sensory input was applied without any retuning to the real robot. We only zero-meaned the gyro rates, by computing the average value over 5 seconds with the robot not moving and using this value as zero reference. All the other parameters (filters parameters, neural-network weights, convergence parameters of the CPG etc.) were kept as in the simulator. The experiment was similar to the previous one except that the robot was placed on a plank which laid on top of a hemisphere, allowing it to be rotated in any direction. The platform was moved by hand performing

a sequence of discrete and close to rhythmic motions. Direction, speed and amplitude of the motion were thus close to random. The learned controller performed very well at stabilizing the robot both around the rolling and pitching axes and with a combination of both (as in simulation). The robot was able to stand on the platform without falling for most experiments, for at least 40-50 seconds. However, since the controller uses only gyro rates as sensory feedback, it slightly drifts during the experiment which causes the falls. An absolute reference such as the gravity could be used to compensate for this drift.

Figure 5.14 shows snapshots of this experiment. Figure 5.15 shows the commands, encoders and feedback for the left knee joint. The first thing worth noticing is that even though the feedback generated by the neural network (Figure 5.15b) is not very smooth, the commands output by the CPG (Figure 5.15a) are. The motor encoders follow the required trajectories well, but the absolute encoders differ substantially. This is due to the effect of the springs. Please also note that the distance between the motor encoder and the absolute encoder is non constant, which shows the non-linear effect of the springs. Thus the effect of the springs is important and should not be neglected. Figure 5.16a shows the pitching and rolling angles of the trunk when the robot is not actuated and subject to similar platform rotations as when it is actuated. The robot was held manually to prevent it from falling. When actuated (Figure 5.16b) the robot is able to significantly reduce its pitching and rolling rotations and to stay stable on the platform without any help. Also note that the movements of the center of mass of the robot (5.16c), although not part of the learning process, are significantly reduced by the method (5.16d). Since we only use gyro rotational rates as sensory input and no general reference, the position of the COM slowly drifts from its original position (red dot in Figures 5.16c and 5.16d), which eventually leads to the robot falling (usually after more than one minute on the moving platform). This issue could be accounted for by adding a small feedback term to the gyroscope input depending on the absolute trunk orientation provided by the IMU, or by using the COM speed as input to the neural network.

5.7 Conclusions

In this chapter we presented two approaches to use vision, instead or in addition to a gyroscope, to control the posture of a compliant humanoid standing on a platform. The first approach is kinematic based, and ensures that the trunk stays straight up while the feet do not slide on the platform. The second one is a model-free controller based on a neural network fusing sensory information and mapping it to joint velocities. Joint integrators then output joint positions to the robot. We systematically tested these two approaches for different frequencies of the pitching and rolling motions of the platform. Both the model-based and the model-free approaches improved the performance compared to not controlling the robot. While the model-based approach was successful at reducing especially the rolling angle of the robot, it showed limitations when reaching higher frequencies. In contrast the model-free approach, when properly trained, improved both the time to fall and the rolling and pitching angles by a great

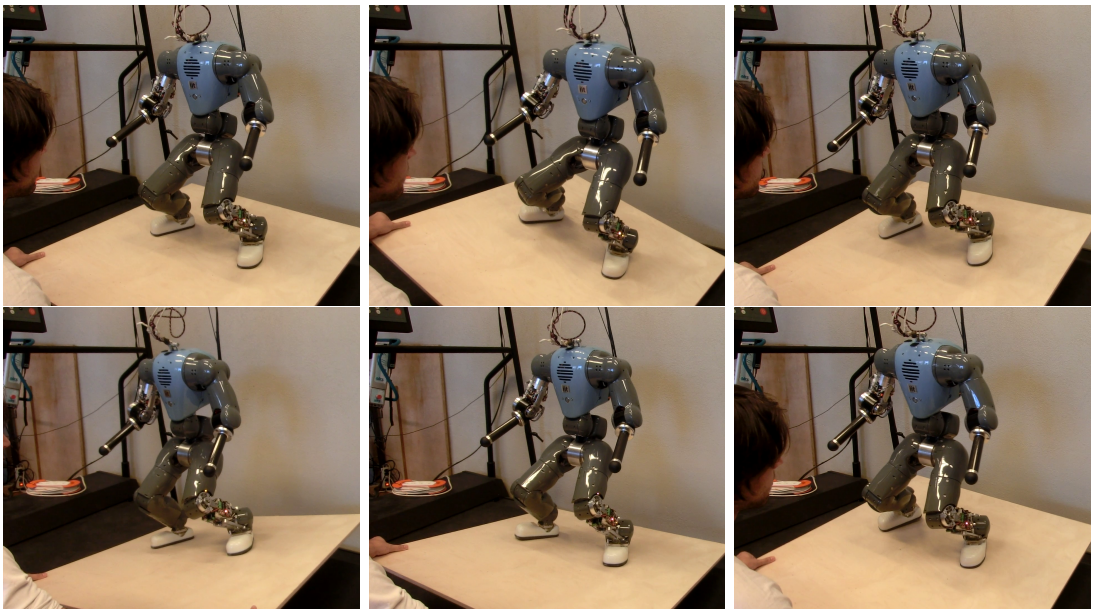
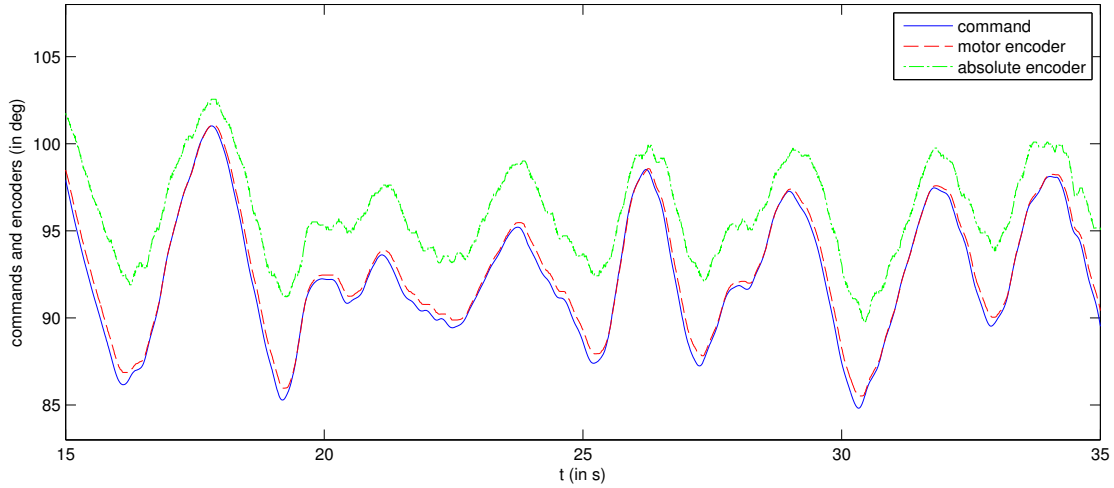


Figure 5.14: Snapshots of the model-free approach applied on the real Coman robot. When the platforms moves the trunk of the robot stays roughly upright.

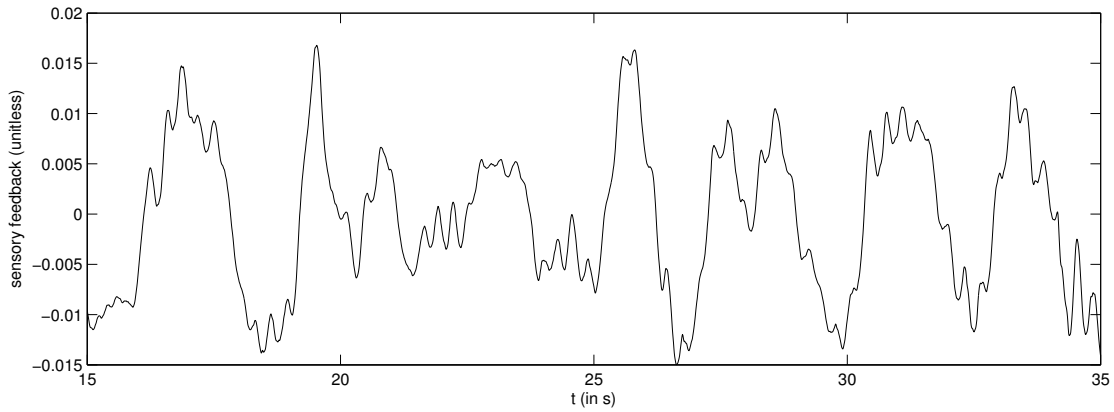
amount. This proves that a linear kinematic control is not sufficient for controlling the posture of a compliant robot. The superiority of the neural-network approach over the model-based one can be explained by the fact that it maps sensor values representing rotational velocities of the robot, in contrast to the model-based control which relies on absolute rotation angles. Thus the neural network can in fact model some aspects of the dynamics of the robot and deal better with its compliance and inertia.

By describing what and how to modulate the parameters of the low-level integrators, we addressed question B while by describing our learning framework we addressed question C, i.e. how to interface sensors with the locomotion controller. We also investigated how to fuse vision and vestibular sensors (question D) and how to use the framework to control a compliant robot (F).

Both model-based and model-free approaches were slightly more successful when using vision than when using the gyro. One reason for that could be that the visual estimation of the rotations of the robot is less sensitive to noise since it integrates a great number of optical flow vectors, while the gyro provides only three values. Furthermore, the fact that stabilization is possible using the model-free approach with raw optical flow as input and no rotation estimation is an interesting feature. Fusing gyro and optical flow together significantly improved the success rate compared to using only the individual sensors.

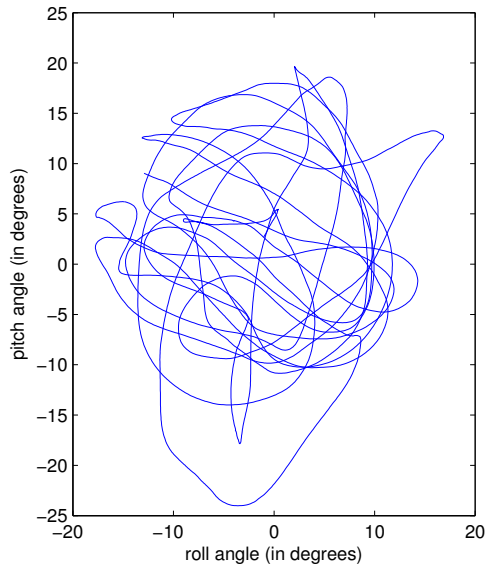


(a) commands and encoders for the left knee joint

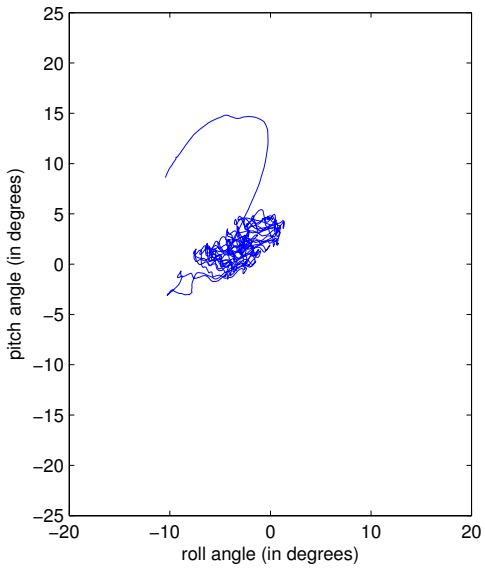


(b) feedback for the left knee joint

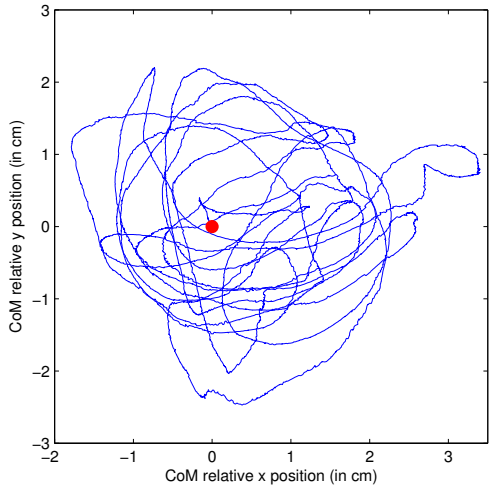
Figure 5.15: Data for the experiment of stabilizing the real Coman robot using the model-free approach. Even though the feedback (b) generated by the neural network is not very smooth, the commands (a) are smooth. The absolute encoders (after the spring) show the non-negligible effect of the spring.



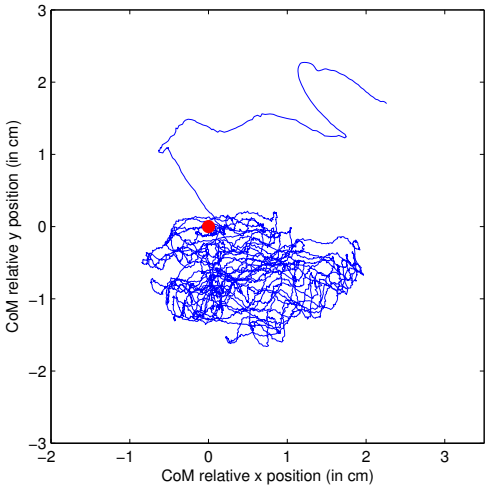
(a) pitch and roll angles of the trunk when the robot is not actuated



(b) pitch and roll angles of the trunk when the robot is actuated



(c) x and y position of the COM when the robot is not actuated



(d) x and y position of the COM when the robot is actuated

Figure 5.16: Performance on the real Coman robot using the model-free approach. The approach reduces significantly the pitch and roll angles of the trunk (b) compared to when the robot is not actuated (a). Even though the center of mass is not explicitly used in the learning process, its deviation from the initial posture (c and d, red dot) is reduced by the system, resulting in the robot keeping balance. However, it slowly drifts away from its original position which causes the robot to eventually fall.

The model-free approach was also tested against discrete platform movements. It proved very successful although the training process involved only periodic movements (sine wave). This proves that our controller learned a rather complete model of the robot and the mapping between sensor and actuators.

Note that our work does not mean to imply that all model-free methods are better than kinematic based methods. We presented here one model-free method versus a relatively classical kinematic based one, and thus can only draw conclusions on these two. We believe both approaches have interesting features which make them better suited to different situations. When an kinematic model of the robot is available, implementing the model-based approach might be the best way to get stabilization working fast. Besides, a kinematic model might be more general as less work should be needed to adapt it to new tasks. For instance, applying the model-free method to a different initial posture would imply retraining the controller, while the model-based method would be directly applicable with few retuning. When dealing with inertia and compliance of the robot is critical, the model-free approach may be better suited.

For future experiments, it would be interesting to check if taking the best individual of the optimization really leads to the best generalization performance. Indeed, it is well known in the optimization community that the very best individual is often over-specialized to its training data. Instead, taking the best individual of previous iterations might show better performance in unseen circumstances. It would be interesting to investigate fusing more sensors with the neural network, like force sensors, and to compare our model-free approach to an inverse dynamics or ZMP based approach.

Chapter 6

Vision for Gait Stabilization

This chapter goes one step further in the design of a controller to learn legged robot stability on rough terrain. It presents a derivation of the system presented in Chapter 4 applied to a compliant quadruped robot, Oncilla, walking on uneven terrain. The problem considered here is an example of a controller based on central pattern generators enabling the robot to walk on flat terrain, which is modulated in order to deal with uneven terrains.

As the previous one, the work of this chapter deals with aspects of the *Sensor Processing*, *Mid Level Control* and *Low Level Control* layers (layers 2, 4 and 5 in Figure 1.3). As Chapter 4 and 5, it addresses questions B and C, i.e. what to modulate in the locomotion patterns in order to achieve the given task and how to interface vision with the lower layers. It also deals with aspects of sensor fusion (question D), here vestibular (gyroscope) and visual (camera) sensors. In this chapter we will use the Oncilla compliant quadruped robot, thus addressing question F.

As for the task considered in Chapter 5, postural control, vision is usually the preferred sensor for gait stabilization, vestibular and force sensors being more commonly used. However, as explained in Chapter 1, vision has a proven influence on the center of mass trajectory during locomotion in humans. Therefore, the goal of this chapter is to investigate whether vision can improve the locomotion performance of the Oncilla robot walking on rough terrain, by replacing or being fused with vestibular sensors.

The work of this chapter can be viewed as an extension of the one of Chapter 5. Indeed, while discrete postural adjustments are still active (i.e. those acting on the offset of the CPG), we also implement here reflexes modifying the periodic behavior of the CPG (i.e. perturbing its limit cycle). While the roughness of the terrain can be viewed as external perturbations, not unlike the platform movements in Chapter 5, the fact that the robot is walking makes the problem harder. First the self-motion of the robot makes the sensor readings more difficult to interpret. Second, the response of the robot dynamics to reflex movements may be more complex (phase dependent etc.) when

the robot is walking than when it is standing.

This work was published in [40].

6.1 Introduction

Balance control during locomotion is critical for legged robots to move in a rough or dynamic environment. Being able to locomote next to a human wherever he/she goes while ensuring the robot and human integrity is a challenge that remains unsolved to this day. One of the precursor of this field is the MIT LegLab which has produced self stabilizing monopod, biped and quadruped robots([88], [89]). The control behind this self-stabilization was composed of a small set of simple PID-like laws. An extension of this work has been implemented in the Big Dog robot from Boston Dynamics which achieved impressive results at walking outdoors in rough environments ([87]). This approach however is very specific and relies strongly on powerful actuators, precise sensors and accurate state estimation.

Another well known approach in the field is the crossing of very rough terrains with the robot Little Dog, also from Boston Dynamics [75]. In [20], a model of the world was extracted from external cameras, and a footstep planner was in charge of finding an achievable path from one end of the terrain to the other. The robot then achieved that path by generating motions using inverse models. The robot was successful at crossing this kind of very rough terrain, but its movements were basically a succession of discrete movements for precise foot placement. The robot was always in a statically stable posture, and had perfect knowledge of the whole environment.

In this paper we also want to allow a four legged robot to cross a rough terrain, but we want to do so while keeping a dynamic gait. One popular line of work aims at decoupling the rhythmic motion and the feedback, by using central pattern generators (CPG), i.e. network of coupled oscillators, as an open-loop controller, which is then modified by sensory feedback. Sensory feedback can be readily integrated in CPGs. In [78], the oscillators are decoupled and feedback from force sensors actually generates the inter-limb coordination. In [27], two very basic feedback loops implementing contact feedback and phase resetting for swing stance transitions have shown to increase the robot stability on slopes.

In [24], robot stabilization was achieved on the Tekken Robot controlled with a CPG by implementing a set of reflexes inspired by biology inside the equations of the CPG. An extension of this work, with an aim at biologically plausible control was presented in [69]. The controller was composed of different kinds of neurons, some responsible for the generation of the rhythmic joint trajectories for the legs and others responsible for shaping these trajectories using sensory information. The feedback functions were also inspired from biology.

Implementing feedback in CPGs has recently been investigated on the HyQ robot ([97]). In [11] the authors use a kinematic model of the robot to adjust the foot locus according to the robot orientation, and inverse dynamics to stabilize it on challenging terrain. However, the feedback is simply superimposed to the trajectories generated by the CPG, and thus the stability properties of CPGs are not exploited.

In [7] and [6] reflexes are integrated inside a CPG enhanced using virtual model control. This time kinematic information is used to include feedback inside the CPG dynamics. The trajectories with sensory feedback corrections are directly generated by the CPG.

Our work is in a sense similar to these latest lines of work. We use a central pattern generator to generate the rhythmic motion for the legs and modify this CPG using sensory feedback so as to stabilize the robot when walking on rough terrain. The main difference between our approach and the pieces of work presented before is that we do not explicitly define the feedback functions modifying the CPG. Instead we want the robot to learn how to modify its own CPG using its sensors in order to maintain balance. Here we choose to use the gyroscope velocities and the optical flow from the camera as sensor information. As both these sensors represent speed rather than absolute orientations, they provide precious information about the robot dynamics. To enable this learning of stability we choose to represent the mapping between sensor values and perturbations applied to the radius, phase and offset of the CPG using an artificial neural network (see Chapter 4). The weights of this neural network are then optimized in simulation using particle swarm optimization (PSO).

The choice of CPG for the generation of rhythmic motions was motivated by a crucial property: its global stability (with finite time perturbations) which ensures the smoothness of the generated trajectories. The limit cycle of the CPG that we use is globally stable, causing a trajectory modified by feedback for some amount of time to return to the limit cycle when the feedback disappears. The other main interest of CPG for this work, as mentioned in Chapter 4, is that it decreases the control problem dimensionality to a small set of variables, and thus the number of parameters of the neural-network to optimize. Moreover, we chose to learn gait stability here rather than explicitly writing the equations for it, so as to investigate if optimization could find different strategies to what a researcher would implement, or even what animals would do. To our knowledge, this work is the first to attempt to learn a feedback controller for robot gait stability using CPGs in such a direct way. It is also the first to link optical flow with CPGs to control walking robot balance.

The robot used for this work is called Oncilla [102]. It is a quadruped robot mimicking cat properties, with in-series compliance on each knee joint. After describing the framework used to control this robot in open-loop, we show how to include sensory feedback to modify the gait in order to prevent the robot from falling when the terrain is changing, and our learning procedure. We finish by presenting experiments with the

robot in simulation on different terrains and discuss the results.

6.2 Control Framework

In this section, we present our control framework for learning the mapping between sensor values and central pattern generator commands. The goal here is to adapt an already existing open-loop gait to cope with changing terrains. We consider two different kinds of terrains: slopes and randomized height maps, but our control framework could be applied to any terrain in theory. Figure 6.4 shows these terrains in simulation. Our goal is to be able to cross these terrains by slightly modifying our existing gait, and thus keeping a dynamic rhythmic motion, rather than by performing a series of discrete movements for careful foot placement. Since obtaining this open-loop gait is not the main purpose of this paper, we will only briefly describe the methodology in Section 6.2.1. Next we will present how to introduce a neural network as sensory feedback function for the CPG, and the learning procedure.

6.2.1 Open-Loop Central Pattern Generator

The robot is controlled by a network of coupled non linear oscillators, a central pattern generator (CPG). The unit oscillator has been modified from [103]. The general idea of this oscillator is to be able to control the duty factor of the gait - the ratio of the duration of the stance phase and the total stride duration - by applying a skewed sine wave to the protraction-retraction joint of the hips. Furthermore the shape of the foot locus can be tuned by applying a double peak trajectory to the knee joint, the duration of each peak being defined by the duty factor. The main motivation to use CPGs here is to exploit their natural properties of robustness to perturbations and smoothness, critical features when introducing sensory feedback. The abduction-adduction joint of the hips is not used for the open-loop gait, and only for discrete movements using feedback. The main difference between the oscillator used here and the one in [103] is that we use a Hopf-like convergence behavior for the amplitude of the oscillator, which is useful when introducing feedback. The equations of the unit oscillators used for the hip and knee are given below:

$$\dot{r}_h = \gamma(\mu_h - r_h^2) r_h \tag{6.1}$$
$$\dot{\phi}_h = \omega \tag{6.2}$$
$$\theta_h = r_h \cos(\phi_L) + o_h \tag{6.3}$$

where ϕ_L is a filter applied on the phase given by:

$$\phi_L = \begin{cases} \frac{\phi_{2\pi}}{2d} & \text{if } \phi_{2\pi} < 2\pi d \\ \frac{\phi_{2\pi} + 2\pi(1-2d)}{2(1-d)} & \text{otherwise} \end{cases} \tag{6.4}$$
$$\phi_{2\pi} = \phi \pmod{2\pi} \tag{6.5}$$

$$\dot{r}_{k1} = \gamma(\mu_{k1} - r_{k1}^2)r_{k1} \quad (6.6)$$
$$\dot{r}_{k2} = \gamma(\mu_{k2} - r_{k2}^2)r_{k2} \quad (6.7)$$
$$\theta_k = r_k\Gamma_k + o_k \quad (6.8)$$

with:

$$r_k = \begin{cases} r_{k1} & \text{if } \phi_{2\pi} < \pi \\ r_{k2} & \text{otherwise} \end{cases} \quad (6.9)$$

$$\Gamma_k = \begin{cases} -16\phi_N^3 + 12\phi_N^2 & \text{if } \phi_N < \frac{1}{2} \\ 12(\phi_N - \frac{1}{2})^3 - 12(\phi_N - \frac{1}{2})^2 + 1 & \text{otherwise} \end{cases} \quad (6.10)$$

$$\phi_N = 2(\frac{\phi_k}{2\pi} \pmod{0.5}) \quad (6.11)$$

r_h and r_k are the radiuses of the hip and knee oscillators, μ_h is the hip target amplitude, μ_{k1} and μ_{k2} the knee stance and swing amplitudes, ω their frequency, ϕ_h and ϕ_k their phases, o_h and o_k their offsets and θ_h and θ_k their outputs. γ is a positive gain defining the speed of convergence of the radiuses to the target amplitudes μ_h, μ_{k1} and μ_{k2}. d is the virtual duty factor, the actual duty factor depending on the robot dynamics and on parameters of the gait. Hip and knee are coupled so that $\phi_k = \phi_h + \psi_{hk}$, where ψ_{hk} is the desired phase shift between hip and knee. Figure 6.1 shows the commands sent to hip and knee for three different values of the virtual duty factor.

The four hips of the robot are also phase-coupled in order to synchronize them, to achieve different gaits. The coupling between hip oscillators i and j is obtained by adding a term to Equation 6.2 as follows:

$$\dot{\phi}_{hi} = \omega + w_{ij}sin(\phi_{hj} - \phi_{hi} - \psi_{ij}) \quad (6.12)$$

where ψ_{ij} is the desired phase difference between the oscillators controlling hips i and j and w_{ij} is a positive gain defining the coupling strength. Figure 6.2 shows the general structure of our CPG.

The different parameters of this CPG (amplitudes, offsets, frequency, duty factor, coupling weights) have been tuned in simulation using particle swarm optimization, with a fitness function aiming at minimizing the pitching and rolling angles of the robot and maximizing the speed of locomotion. These parameters have then been implemented on the real robot for validation and hand-tuned. The obtained gait has then been ported back to the simulator to carry out the work described in this paper. This work should thus easily be transferable to the real platform.

6.2.2 Including Sensory Feedback in the CPG

The main point of the paper is to use the CPG presented in Section 6.2.1 and introduce sensory feedback to enable the robot to adapt to changing environments. Our controller is modular and the CPG remains fully operational if the feedback is disabled.

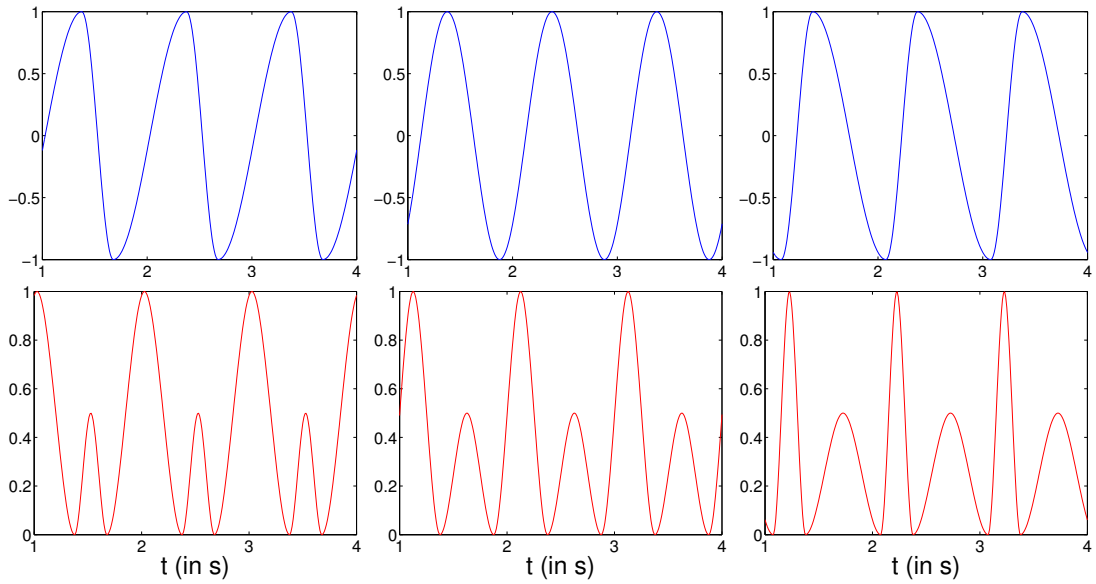


Figure 6.1: The hip (top, blue) and knee (bottom, red) commands for different values of the virtual duty factor: $d = 0.3$ (left), $d = 0.5$ (middle) and $d = 0.7$ (right)

Following the same principle as in Chapter 4, we modify the equations presented in Section 6.2.1 by introducing feedback on the radius, phase and offset variables of the hip oscillator and on the radius and offset of the knee oscillator (the phase being shared with the hip). The new equations for the hip and knee are given below:

$$\dot{r}_h = \gamma(\mu_h + \kappa_r \mathcal{F}_h^r(\mathbf{S}) - r_h^2) r_h \tag{6.13}$$
$$\dot{\phi}_h = \omega + w_{ij} sin(\phi_j - \phi_i - \psi_{ij}) + \kappa_\phi \mathcal{F}_h^\phi(\mathbf{S}) \tag{6.14}$$
$$\dot{o}_h = \kappa_o \mathcal{F}_h^o(\mathbf{S}) \tag{6.15}$$

$$\dot{r}_{k1} = \gamma(\mu_{k1} + \kappa_r \mathcal{F}_k^{r1}(\mathbf{S}) - r_{k1}^2) r_{k1} \tag{6.16}$$
$$\dot{r}_{k2} = \gamma(\mu_{k1} + \kappa_r \mathcal{F}_k^{r2}(\mathbf{S}) - r_{k2}^2) r_{k2} \tag{6.17}$$
$$\dot{o}_k = \kappa_o \mathcal{F}_k^o(\mathbf{S}) \tag{6.18}$$

where:

$$\mathcal{F}_k^{r1}(\mathbf{S}) = \begin{cases} \mathcal{F}_k^r(\mathbf{S}) & \text{if } \phi_{2\pi} < \pi \\ 0 & \text{otherwise} \end{cases} \tag{6.19}$$

$$\mathcal{F}_k^{r2}(\mathbf{S}) = \begin{cases} \mathcal{F}_k^r(\mathbf{S}) & \text{if } \phi_{2\pi} \geq \pi \\ 0 & \text{otherwise} \end{cases} \tag{6.20}$$

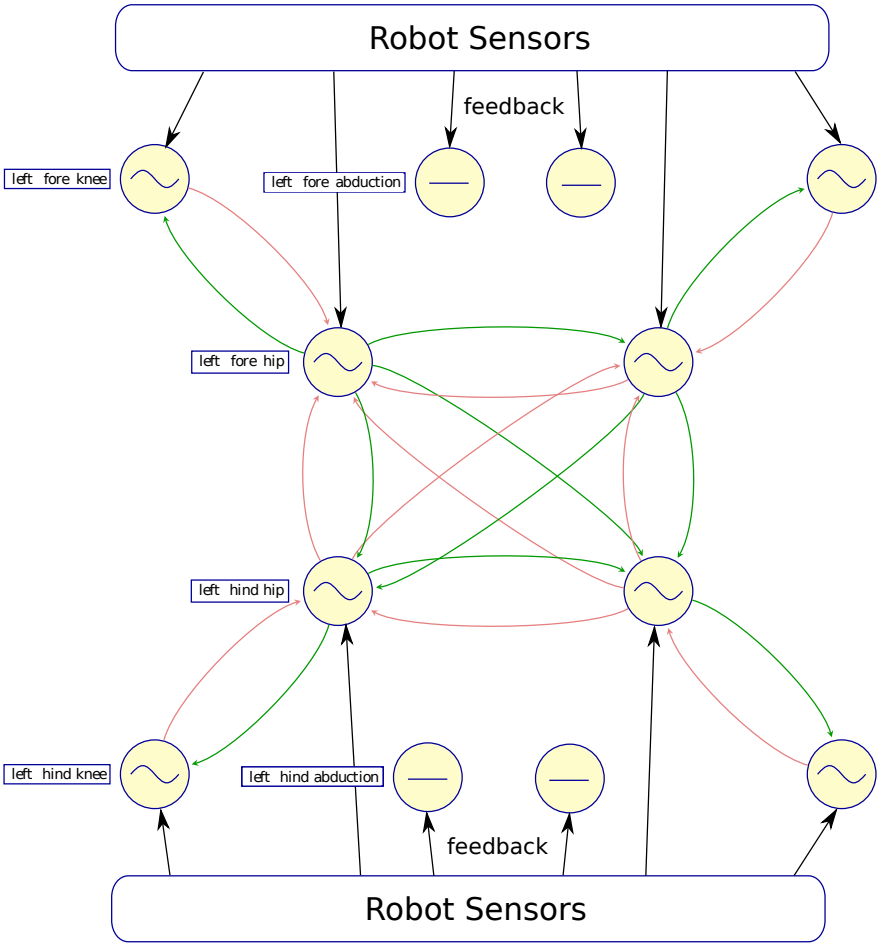


Figure 6.2: The CPG used for the Oncilla robot. The hip and knee joints are controlled by the oscillators presented in Section 6.2.1 and coupled. The ablation joints are idle in the open-loop case, and only perform discrete movements controlled by the simple integrator described in Section 6.2.2. All oscillators and integrator accept sensory feedback which modify their outputs.

The abduction/adduction joints are also used to exploit the sensory feedback and increase the robot stability. They are decoupled from the other joints and perform only discrete movements given by the following equations:

$$\dot{r}_a = \gamma(\kappa_r \mathcal{F}_a^r(\mathbf{S}) - r_a^2) r_a \quad (6.21)$$

$$\dot{o}_a = \kappa_o \mathcal{F}_a^o(\mathbf{S}) \quad (6.22)$$

$\mathcal{F}_h^r$, $\mathcal{F}_h^\phi$, $\mathcal{F}_h^o$, $\mathcal{F}_k^r$, $\mathcal{F}_k^o$, $\mathcal{F}_a^r$, $\mathcal{F}_a^o$ are functions of the sensor values $\mathbf{S}$ to be defined in Section 6.2.3., and κ_r, κ_ϕ and κ_o are positive scaling factors. As described in Section 6.2.1, the oscillator controlling the knee has two radiuses, r_{k1} for the stance phase and r_{k2} for the swing phase. The feedback for the knee radius is thus decoupled into $\mathcal{F}_k^{r1}(\mathbf{S})$ acting on r_{k1} only during the stance phase and $\mathcal{F}_k^{r2}(\mathbf{S})$ acting on r_{k2} only during the swing phases.

The effect of the three feedback functions on the respective variables and output of the CPG is explained in Chapter 4, Section 4.2.1.

6.2.3 Learning the feedback functions

Now that the low-level CPG has been defined the feedback functions $\mathcal{F}^r$, $\mathcal{F}^\phi$ and $\mathcal{F}^o$ need to be designed such that they map the sensor values to the right CPG modifications to stabilize the robot. As explained in Chapter 4, we decided to represent these feedback functions with an artificial neural network, and particle swarm optimization to tune its weights.

Figure 6.3 shows the full control framework including the learning process. As in Chapter 4, the first step is sensor processing. We get rotational speeds from the gyroscope and images from the camera. The optical flow is then computed from the images of the camera and down-sampled by splitting the image (typically in four quarters) and averaging it in each part. The sensor vector $\mathbf{S}$ is thus composed of all or a subpart of the three gyroscope values and $2K$ values for the camera (x and y component of each vector, K being the number of parts the image is split into for the down-sampling).

As in Chapter 4 and 5, theses sensor values are used as input to a fully connected neural network with sigmoid activation, which outputs the values for $\mathcal{F}^r(\mathbf{S})$, $\mathcal{F}^\phi(\mathbf{S})$ and $\mathcal{F}^o(\mathbf{S})$ for each hip joint, and $\mathcal{F}^r(\mathbf{S})$, and $\mathcal{F}^o(\mathbf{S})$ for each knee joint and abduction joint. The robot having 4 hip joints, 4 abduction joints and 4 knee joints, the total number of outputs of the neural network is 28. The CPG takes these functions as sensory feedback and outputs joint positions for each joint. The weights of the neural network W^D are tuned by an reward-based learning process, using particle swarm optimization (PSO). Together with the weights of the neural network, convergence parameters of the CPG and the slope of the sigmoid of the neurons are tuned since they also determine the influence of the feedback on the output trajectories. The total parameter vector

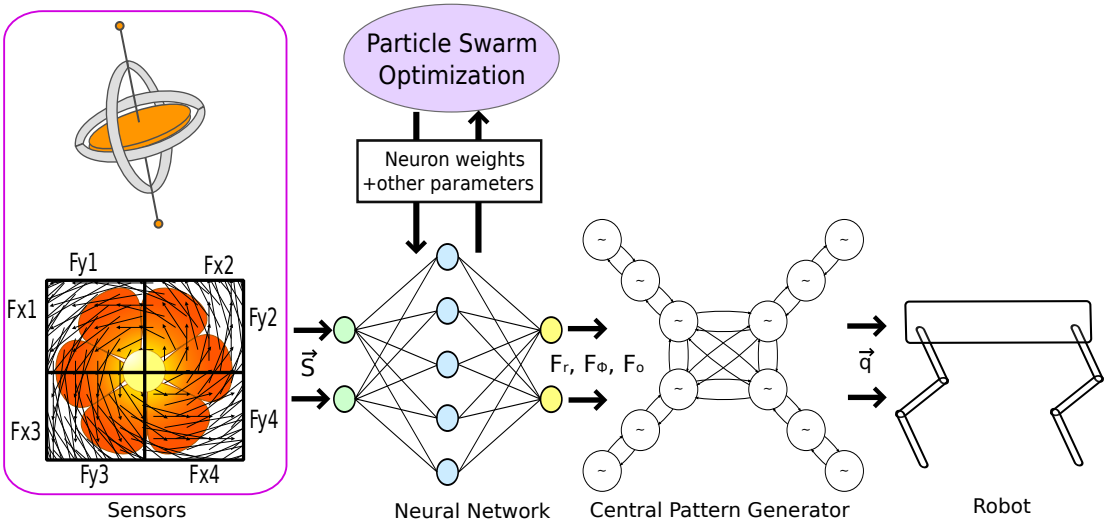


Figure 6.3: General idea of our framework for learning stability. Sensor values are processed from various sensors. Optical flow is computed from the images of the camera and averaged in four quarters of the image. The sensor values are then fed to a fully connected neural network which weights are optimized using particle swarm optimization (PSO). This neural network outputs the feedback values for the CPG controlling the robot

for the optimization is: $[W^D, \gamma, w_{ij}, \kappa_r, \kappa_\phi, \kappa_o, \lambda]$, λ being the slope of the sigmoid $\frac{1}{1+e^{-\lambda x}}$

A typical optimization scenario is then to initialize a first random population of these parameters, run the simulation with the type of terrain considered, and record a measure of fitness for the optimization process. Here we use the fitness function described in Chapter 4, Section 4.3 which aims at maximizing the traveled distance but also at minimizing the integrated pitch and roll angles of the robot during the simulation.

This fitness is used by the optimization process to generate the next populations by selecting the best individuals. This process is quite heavy computationally (typically more than 100 iterations to converge, with 100 particles per iteration). However, once learned, the feedback controller only requires a simple feedforward neural network computation, and is thus much cheaper computationally than methods based on inverse models.

6.3 Experiments

In this section, we present experiments we performed on two different terrains: a descending slope and a height map simulating a random rough terrain. Figure 6.4 shows the kinds of terrains we are considering. For all the experiments described next, the

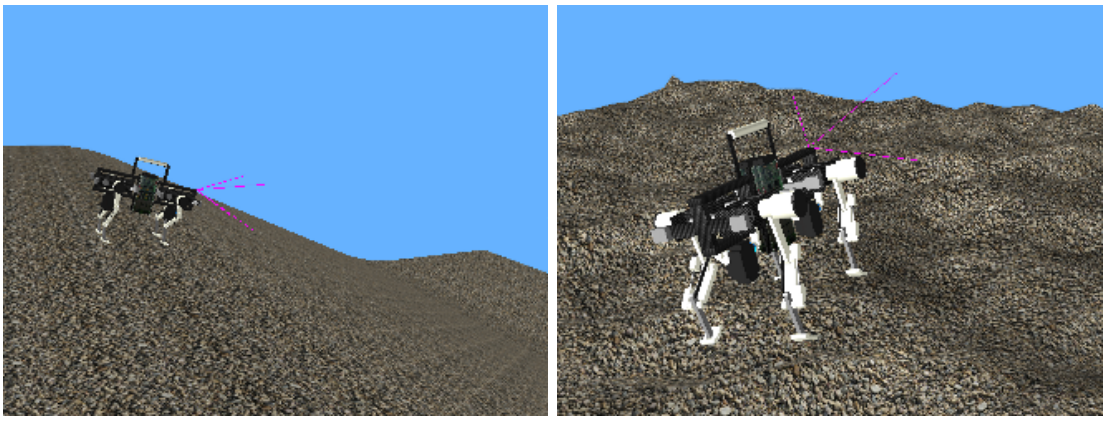


Figure 6.4: The two worlds used for his work. Left: a slope. Right: height map

open-loop gait is the same. We use a frequency of 2.5Hz, amplitudes for the hip protraction/retraction of 15 degrees, amplitude of the knees of 0.5 and 0.05 (unit-less, an amplitude of 1 meaning full flexion) for the swing and stance respectively, and a duty factor of 0.6. This gait is used both in simulation and on the real robot, with similar performance. It is a very dynamic gait, where the springs are used extensively and which reaches about 40 cm/s (1.7 body-lengths/s) both in simulation and on the real Oncilla robot. We first show the results when learning in each terrain, and analyze the strategies found by optimization. Then we investigate if the feedback functions learned on one terrain improve the performance of the robot on the other. All simulations are performed using a model of the Oncilla robot in Webots ([70]), a robotics simulation software based on the Open Dynamics Engine (ODE). A video showing the results of this work is available with this paper and a better quality version can be watched in [1].

6.3.1 Learning Procedure

The learning procedure is the same for each terrain. A first population of parameters are generated where all the weights of the neural network are close to 0, and thus the performed gait is very similar to the open-loop gait. This gives a first good guess to the optimization, where the robot at least moves forward in a stable way on the flat section before the changing terrain. For the next generations, the particles explore the search space as specified by the PSO algorithm. We run the optimization for 300 iterations with 100 particles in each iteration. For each terrain we ran 2 times this optimization procedure: first using only the gyroscope as sensory input and then using only the camera. The camera sampling rate was set to 50Hz, while the gyro sampling rate was set to the control frequency: 167Hz (6ms timestep). Here we want to investigate the performance using each sensor separately, but our idea for future works is to fuse the different sensor inputs, by simply adding them as input to the neural network. We repeated each optimization 3 times for each terrain and each sensor to check that our learning procedure is independent of initial conditions of the particle swarm optimization.

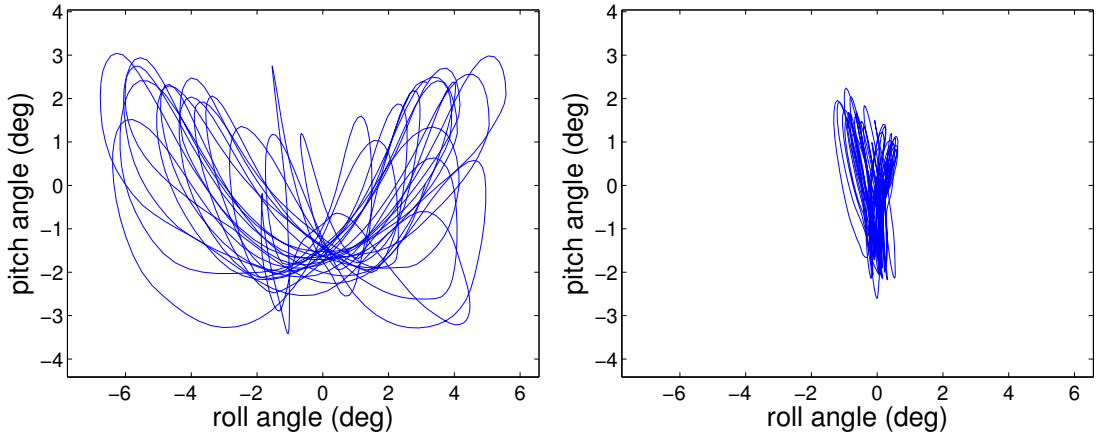


Figure 6.5: Pitch vs Roll angle (in deg) over 10 seconds of simulation with the open-loop (left) and closed-loop (right) controllers on flat ground.

Only one of these repetitions is shown here, but qualitatively similar results (with slight differences in the strategies found) were achieved in each case.

6.3.2 Flat ground

As a first proof of concept, we want to check if our controller can decrease the pitch and roll angles of the robot walking on flat ground. We run the learning procedure described before with the fitness presented in Equation 4.9 (Chapter 4 Section 4.3).

After about 150 iterations the optimization has converged and the resulting controller is able to reduce the average pitching and rolling angles of the robot by 32% and 89% respectively. Figure 6.5 compares the pitch and roll angles of the robot using the open-loop controller and the closed-loop controller.

6.3.3 Slope

The robot is placed in front of a slope (Figure 6.4, left). For each particle of the optimization, we repeat the simulation with 3 different slopes of angles 0.3, 0.4 and 0.5 radians (∼17, 23 and 29 degrees). The fitness of each particle is then computed as the average of the traveled distance on these 3 slopes, including the flat sections before and after the slope. This corresponds as setting β_P and β_R to 0 in Equation 4.9 (Chapter 4 Section 4.3). We do not want to minimize the pitching and rolling angles here, as walking on a slope with a flat trunk is very unnatural and may not lead to the best performance. The idea is that, after learning, the controller should be able to generalize to any slope between these values.

The evolution of the fitness during the optimization using the gyroscope as only sensory input is given in Figure 6.6. The best individual reached a traveled distance of about 3.9 meters which is the maximal achievable fitness (it corresponds to all 3

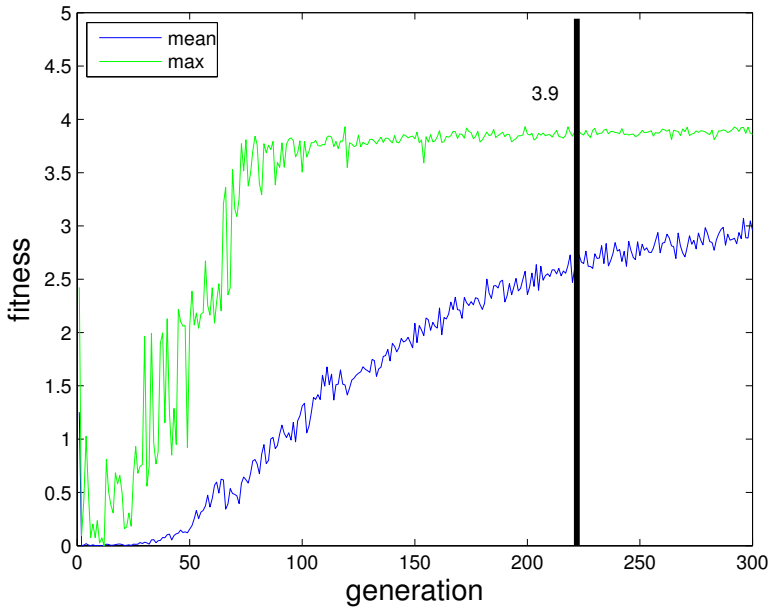
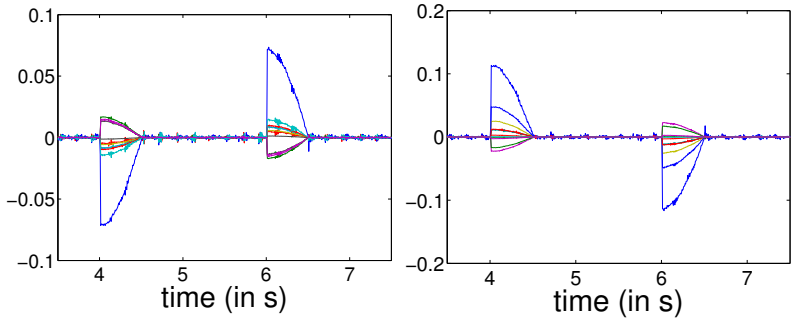


Figure 6.6: Evolution of mean, maximum and minimum fitness (the traveled distance) of each iteration throughout the optimization, using gyro as sensory input.

slopes being crossed). This is qualitatively similar when using the camera, instead of gyro, although the optimization takes longer to converge (about 220 iterations), most likely because of the higher number of parameters (we use 8 values for the optical flow, and only 3 values for the gyro). The similar performance of the controller using optical flow and gyro is an interesting result, since the gyro provides information more than 3 times more often than the camera. Moreover, the information provided by the gyro (rotational speeds of the trunk) is much more explicit than the optical flow provided by the camera, which only gives the linear speed of the pixels. However, the optimized neural network seems to interpret the optical flow as well as the gyro rotational speeds.

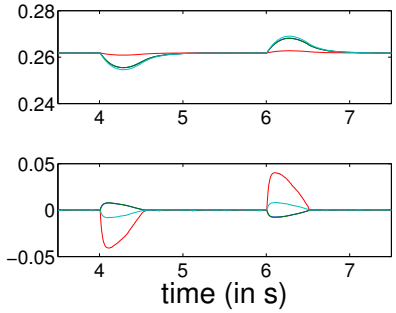
Despite the relatively high number of parameters, (in this case 378), the optimization converges nicely in about 100 iterations. The average fitness keeps getting better trough the rest of the optimization and our best controller emerges after 221 iterations.

It is hard to see what kind of feedback is learned when looking at the data from the robot walking, since the feedback is basically always active. To analyze this feedback, we actuate the robot in the air. To simulate the transitions from flat ground to slope and back, at $t = 4s$ we rotate the robot around its pitch axis for $0.5s$, until it reaches an angle of 0.4 radians (23 degrees). At $t = 6$ we rotate it back to its initial position for $0.5s$. Figure 6.7 shows the feedback on the radius and offset when using gyro as sensory information, as well as the evolution of the radius and offset and the commands of each oscillator. The feedback learned did not make use of the feedback on the phase.

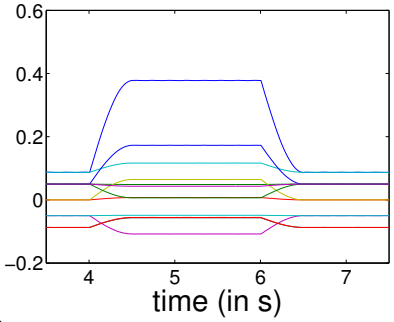


(a) feedback on the radius

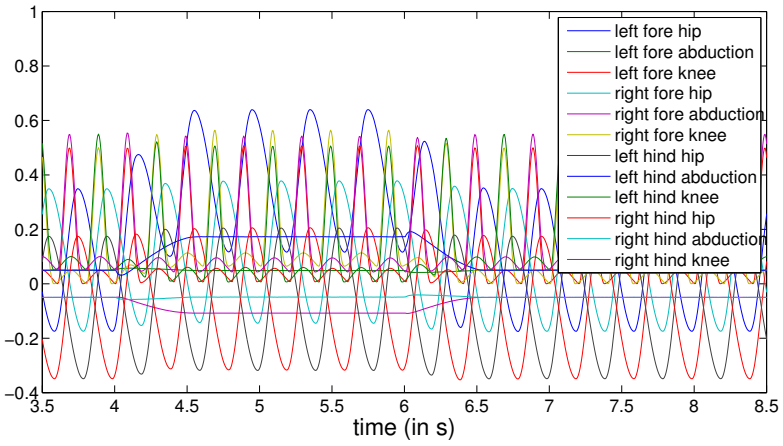
(b) feedback on the offset



(c) evolution of the radius (top: hip protraction, bottom: hip abduction)



(d) evolution of the offsets



(e) commands in radians for hips protraction and abduction, unit-less for the knee (0: knee fully extended, 1: knee fully flexed)

Figure 6.7: Data when the robot is actuated in the air, using gyro as sensory input. At $t = 4s$ the robot is rotated by 0.4 radians. At $t = 6$ it is rotated back by -0.4 radians.

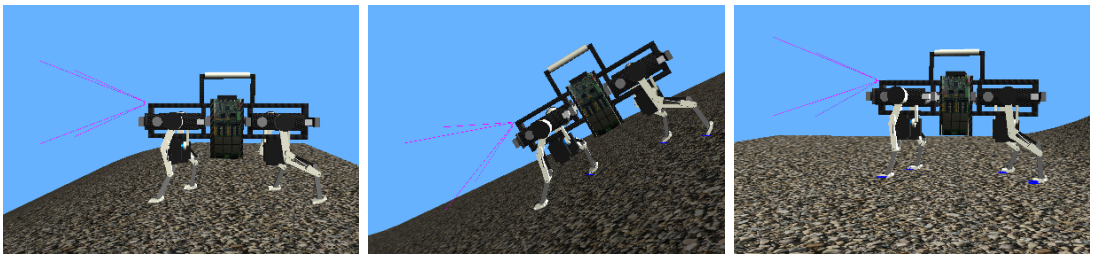


Figure 6.8: Snapshots of the robot walking before the slop (left), on the slope (middle) and after the slope (right). When the robot is on the slope it puts its front left foot forward to compensate its forward inertia. After the slope, it goes back to the normal gait.

The first thing worth noting is that even though the feedback is not very smooth (due to noise, low sampling frequency of the sensor etc. - Fig. 6.7a and 6.7b), the commands applied to the joints are smooth (Fig. 6.7e). This shows the smoothing ability of the CPGs in the presence of perturbations (here sensory feedback). Then we notice that the robot makes good use of both the radius and offset feedback. The robot uses the feedback on the radius to temporarily adapt its leg positions and lengths to compensate the forward inertia during the transitions (Fig. 6.7c). Between the two transitions, a new stable gait is reached where only the offsets are different from the open-loop gait. Finally, let us observe that the new stable gait found when the robot is rotated is asymmetric (see Figure 6.7d). The left fore leg is placed much more forward than the other one to prevent tipping over. This is a strategy for the robot to keep balance that is different to what is usually done with model-based control. Yet, it proves effective since it allows the robot to cross a nearly 55% slope. Figure 6.8 shows snapshots of the robot going down a slope, where one can see this strategy in action.

To test whether our best controller can generalize to other slopes, we run 100 times the simulation with random slopes varying from 0.3 to 0.5 radians (about 17 to 29 degrees). To check that our feedback control is independent of the timing at which the robot arrives on the slope, we also set the initial position of the robot to a random value 10 cm around the position used for the training. We also use longer slopes than in the optimization phase (5 meters instead of 4). For each run, we compare the results to those of the open-loop controller. The results are shown in Figure 6.9. In all cases, the closed-loop controller performed equally or better than the open-loop one. The closed-loop controller was successful in crossing the whole slope in 85% of the cases, while the open-loop was successful only in 26%. Note that the open-loop controller was able to cross slopes up to 19 degrees (about 34%) in all cases, and up to 21 degrees (about 38%) in specific cases. This performance is very good for an open-loop controller, which shows that compliance and finely tuned gait parameters can lead to good open-loop performance on rough terrain. However, the closed-loop controller was able to cross significantly steeper slopes, up to 28 degrees (55%). Figure 6.9 also shows

qualitatively similar behavior of the controller when using camera input instead of gyro input, even though the overall performance was slightly lower when using the camera.

6.3.4 Height Map

We used the same optimization procedure as for the flat and slope grounds on a random height map (random uneven terrain), with the fitness described in Equation 4.9 (Chapter 4 Section 4.3). We repeated the simulation for each particles with 3 different randomized height maps, of respective maximum height of 4, 5, and 6 cm, in a grid of 10cm steps. This corresponds respectively to about 22%, 28% and 33% of the leg length of the robot, and respectively 40%, 50% and 60% maximum local slope.

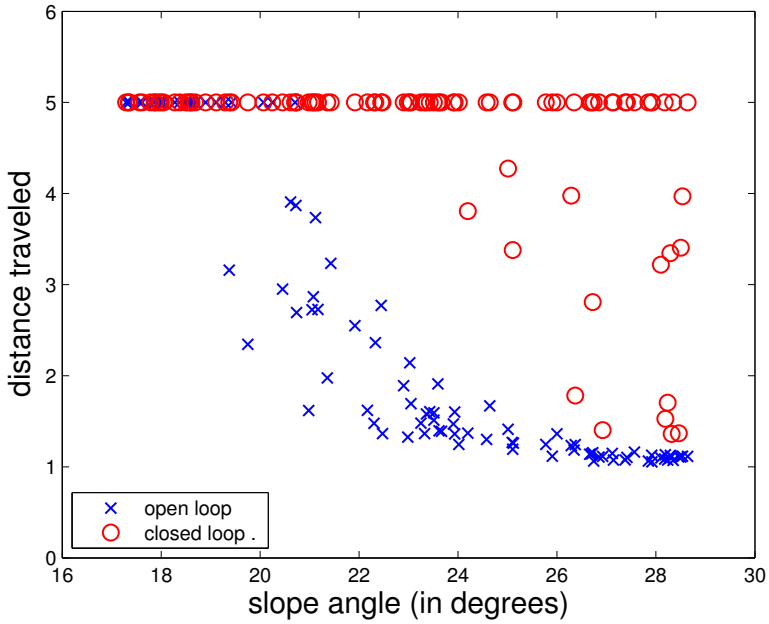
Figure 6.10 (left) shows the evolution of the fitness throughout the optimization. As for the slope, the optimization converges in about 100 iterations. At iteration 80, the optimization has already found a controller able to cross all three worlds. Even after the distance traveled has been maximized, the fitness is further improved by decreasing the pitch and roll angles of the robot, as shown in Figure 6.10 (right).

We ran 100 generalization tests with random height maps of maximum heights ranging from 0.04 to 0.06. Figure 6.11 shows the results. When setting the success distance to 2 meters, as in the learning, the success rate of the closed-loop controller was 57% against 25% for the open-loop controller. The closed-loop controller was better in 74% of the cases with an average traveled distance increase of 41%. When increasing the success distance, the absolute success rates of both the open-loop and the closed-loop controller decrease. However, the performance of the open-loop controller drops faster than the closed-loop one. With a success distance of 2.6m, the performance of the closed-loop and open-loop controllers are 49% and 13% respectively, and 40% and 9% for 2.9m.

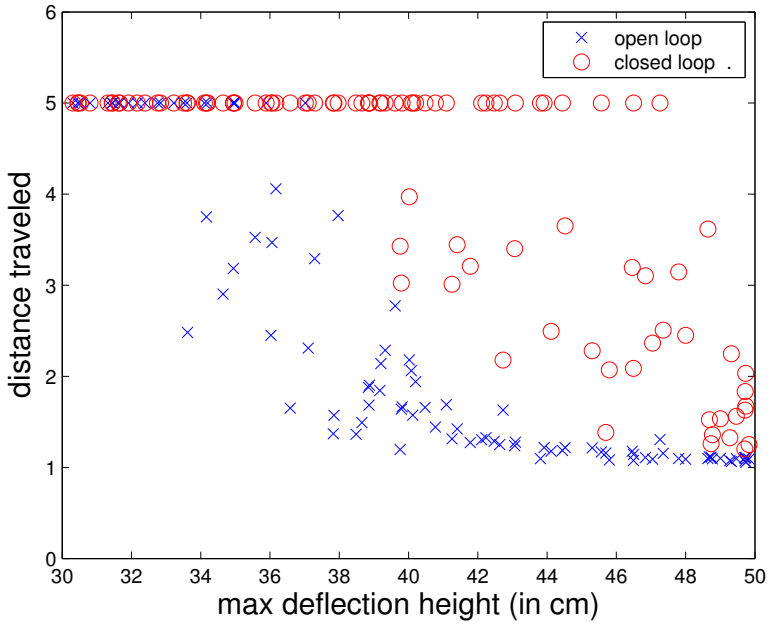
Thus our controller improves significantly the performance of the open-loop controller on height-maps, but is less efficient than on slopes. This can be explained by the fact that the randomized height map contains much more stochasticity than the slope. The large difference between the training and testing set performances implies that our learning process has overfitted to the three training worlds. Thus, simply repeating the learning in only three different worlds might not be enough and a more complex feedback mechanism might be needed to increase the generalization performance.

6.3.5 Phase-dependent feedback

Modulations of the central pattern generator of animals are known to be at least partially phase-dependent ([32], [111]). In this section we investigate if adding phase-dependent feedback can improve the performance of the controller. To learn phase-dependent feedback, we simply add the phase of the CPG (modulo 2π and normalized between



(a) gyro input



(b) camera input

Figure 6.9: Test of generalization of the best learned feedback controller on slopes with gyro input (top) and camera input (bottom). The qualitative performance is similar in both cases, even if the overall performance with camera input was slightly lower in this case (more failures with steep slopes). The maximum achievable distance in this world is 5m, which means any lesser traveled distance implies that the robot has fallen.

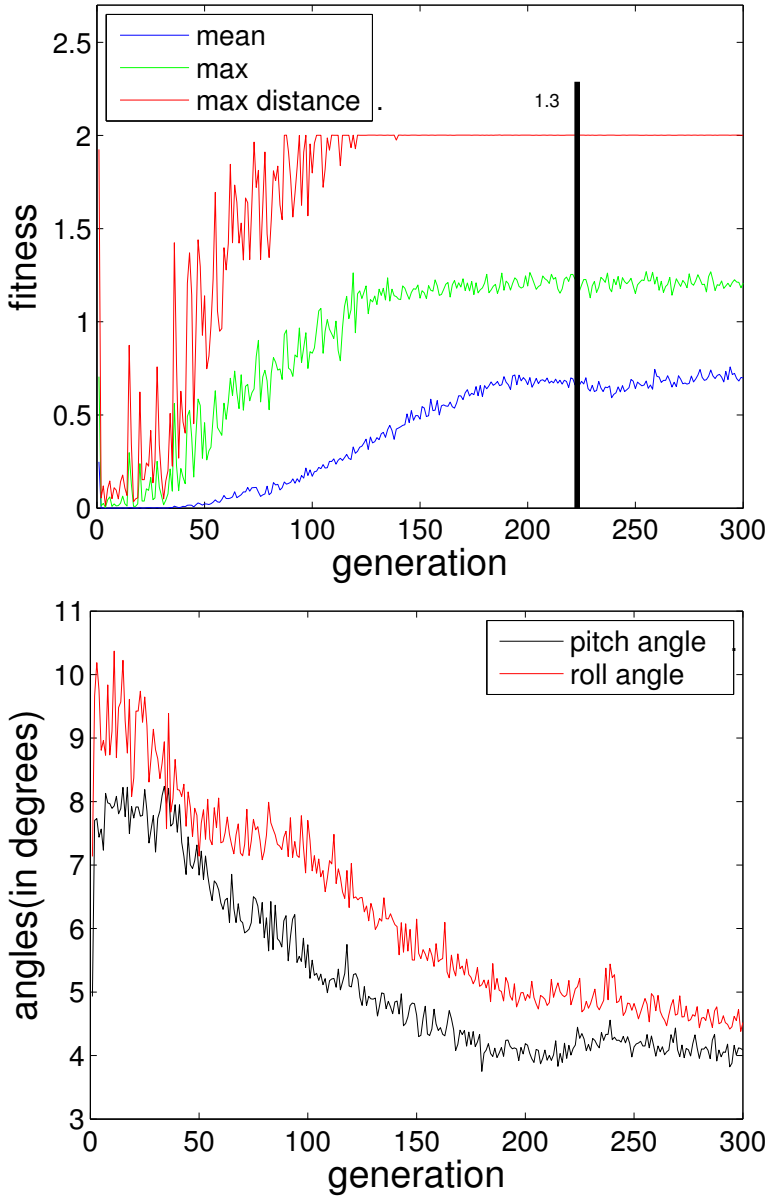


Figure 6.10: Evolution of mean and maximum fitness (left) and the mean integrated pitch and roll angles (right) of each iteration throughout the optimization .

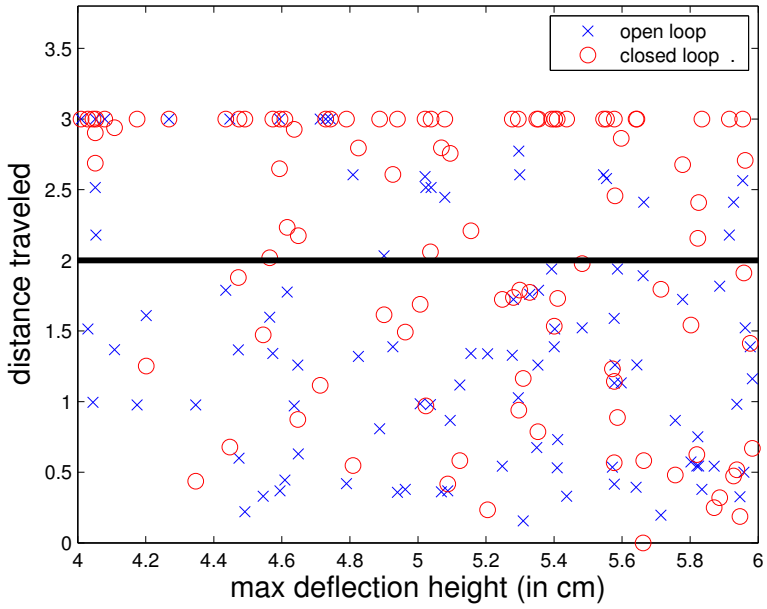


Figure 6.11: Test of generalization of the best learned feedback controller on random uneven terrain. The maximum success distance is set to 2 m.

-1 and 1) as additional input to the neural network. We then relaunch the learning procedure and the generalization tests, as in the previous sections.

Figure 6.12 shows the generalization performance of the controller on slopes and random uneven terrain. Introducing the phase of the CPG did not have any noticeable effect on the general success rate (from 85% success without phase-dependent feedback to 84% with phase-dependent feedback) when walking on slopes. It does however seem to improve the success rate on the steepest slopes. The success rate when considering only slopes higher than 25 degrees (47%) was here close to 77% while it was only 60% when not using the phase as input to the neural network. However, when walking on rough uneven terrain, introducing phase-dependent feedback greatly increases the performance. While without phase-dependent feedback, the success rate barely reached 57%, with phase-dependent feedback it climbs to 77%. The closed-loop controller was in this case better than the open-loop one in 88% of the cases.

This difference of behavior when introducing phase-dependent feedback can be explained by two points. First, the controller without phase-dependent feedback is already close to optimal on slopes. It is always better than the open-loop gait and reaches 85% success, so the margin for improvement is quite low. The second point is the nature of the terrain. On slopes, as shown in Figure 6.7, the controller basically reaches a new steady-state gait on the slope by strongly adapting the offsets before and after the slope. On randomized height maps however, perturbations are so local that

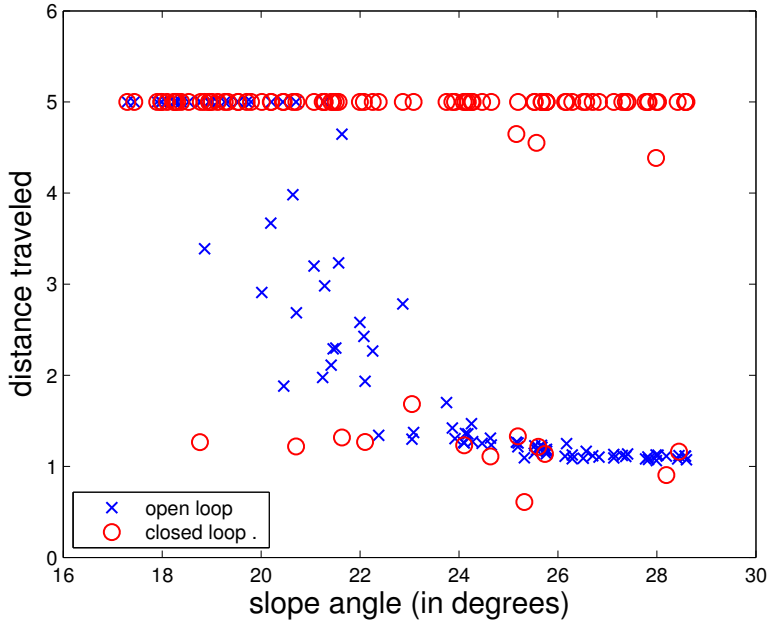
no new steady-state gait can be achieved. The feedback mechanism may need to be more complex than on the slope. Thus providing the phase of the CPG may enable the robot to learn this more complex feedback (allowing it for instance to use different feedbacks during swing and stance to avoid stumbling, similarly to phase-dependent reflexes observed in animals ([33]) and humans ([119])).

We further analyzed how the robot used the phase information to increase the performance on uneven terrain. We applied an impulse feedback on the x gyro coordinate (simulating a forward pitching motion), and recorded the evolution of each variable of the CPG. Figure 6.13 shows the evolution of the offsets on each DoF of the robot. The behavior on the radius is similar and is thus not shown here. The first thing to note is that even in the absence of sensory feedback, the offsets are periodically changed by the phase information provided as input to the neural-network. Thus the robot has used the clock information provided by the phase to design a new open-loop gait, supposedly more stable than the original one. We ran the same systematic tests as previously (Figure 6.12), but this time setting all sensor values to zero and providing only the phase as input to the neural-network. We recorded an overall success rate of 34%. This is significantly higher than the pure open-loop controller (18% on this particular test set), which proves that the controller effectively made use of the phase information to design a new open loop gait better suited to this kind of terrain. Looking back at Figure 6.13, we note that the robot still makes use of the sensor information to implement reflexes and postural control. Indeed, the amplitude of the change in some offsets due to the sensor information is significantly higher than the changes due to the phase.

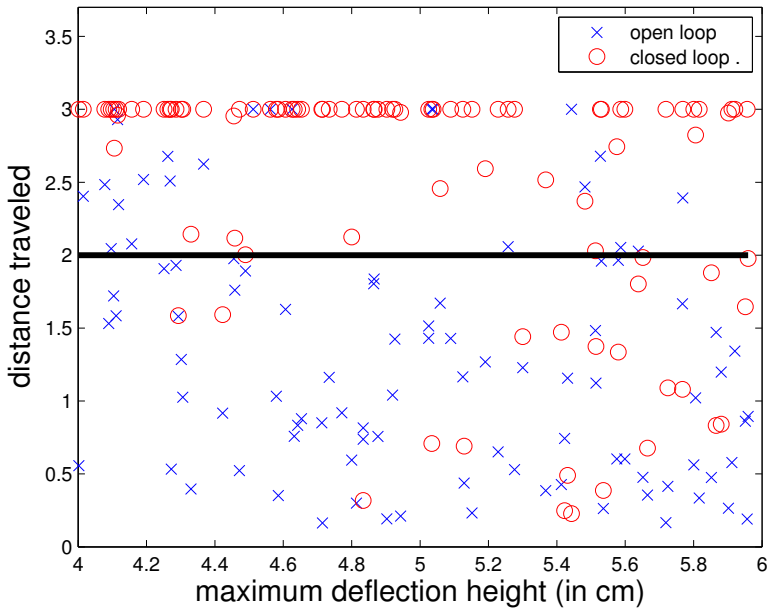
Next, we wish to investigate whether the robot effectively modulates the effect of the sensors according to the phase of the CPG. To do that, we tried providing a fixed feedback to the input of the neural-network corresponding to the gyro x component ($\omega_x = 1$), and different values to the one corresponding to the phase. We fixed the two other inputs (gyro y and z components) of the neural-network to zero and recorded the outputs of the neural network, i.e. the feedback for each variable of each joint. This fixed value of the gyro x component simulates the robot pitching forward. By varying the phase input, we can compute the feedback generated by the neural network when the robot is pitching forward at different points of the locomotion cycle. Figure 6.14 shows that the feedback generated by the neural network indeed depends on the phase. The individual values are not important here. Only note that some values are more influenced by the phase (i.e. have larger slopes) than others, which shows the complexity of the learned feedback mechanism.

6.3.6 Hardware implementation

We started implementing this work on the real Oncilla robot. For now only the open-loop gait has been implemented. The robot reaches a speed of about 40 cm/s ($\sim$1.7 body-lengths/s), which is comparable to what is obtained in simulation. Faster gaits

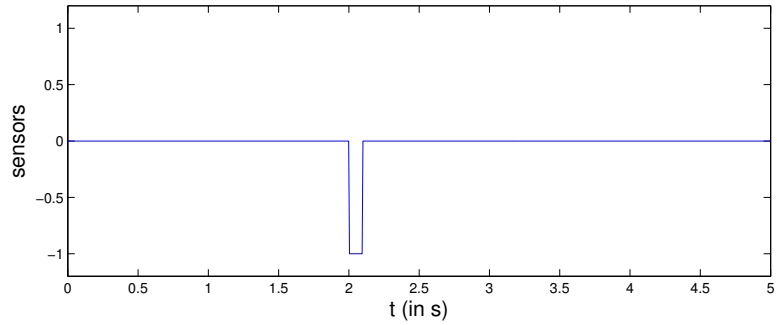


(a) performance on slopes

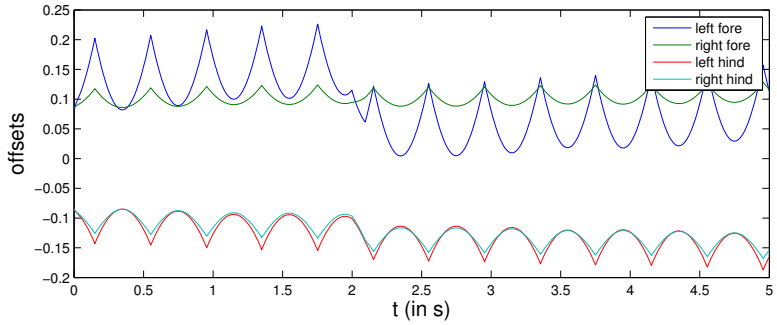


(b) performance on uneven terrain

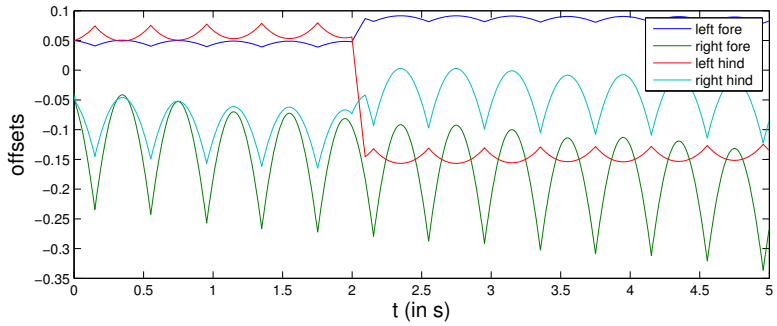
Figure 6.12: Test of generalization of the best learned phase-dependent feedback controller on slopes (top) and randomized uneven terrain (bottom). While introducing phase-dependent feedback actually reduces the performance when crossing slopes, it greatly improves it for random uneven terrain.



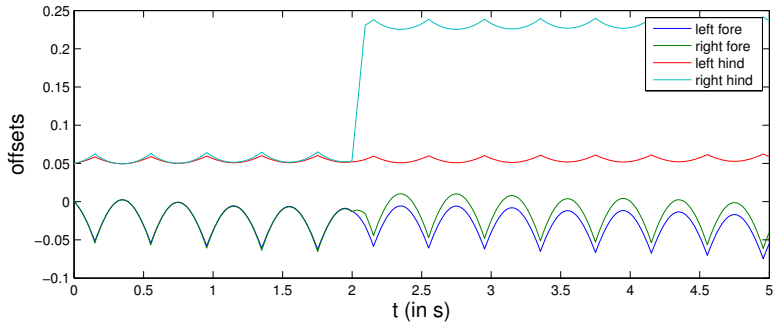
(a) gyro x value



(b) offsets for the hip pitch



(c) offsets for the hip abduction/adduction



(d) offsets for the knees

Figure 6.13: Evolution of the hip pitch, hip abduction and knee offsets when the neural network is provided phase information as input.

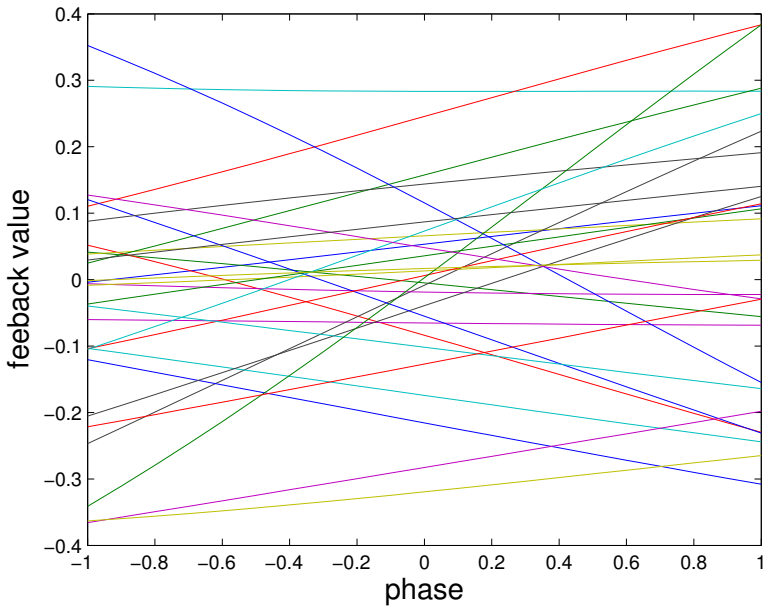


Figure 6.14: Values of the feedbacks generated by the neural network when providing a gyro x input of value 1 and varying the input corresponding of the phase of the CPG.

have also been tested reaching speeds of up to 55 cm/s (∼2.3 body-lengths/s) at 3.5Hz (See movie [1]).

6.4 Conclusions

In this chapter we have applied the framework presented in Chapter 4 to learn gait stability of a compliant quadruped robot. We have shown that by carefully integrating feedback in the CPG, we are able to make full use of its stability properties to generate smooth gait modifications to achieve the considered task. By learning the weights of the neural network, together with the convergence factors of the CPG in a reward-based matter, we showed that the designed sensory feedback can significantly improve the locomotion performance (i.e. better balance, fewer falls) of the robot on rough terrain. We showed that our feedback controller can be used seamlessly with different sensory information, such as the rotational speeds provided by the gyro, but also less explicit information like the optical flow. Our controller does not need any inverse model of the robot, whether kinematic of dynamic, but directly learns the mapping between sensors and gait corrections (note that it does need a forward model, i.e. a simulator). Thus it can be in theory applied to any robot, even robots where obtaining an inverse model is very hard or even impossible like deformable or tensegrity robots.

By describing what and how to modulate the parameters of the low-level CPG, we addressed question B while by describing our learning framework we addressed question

C, i.e. how to interface sensors with the locomotion controller. We also investigated how to fuse vision and vestibular sensors (question D) and how to use the framework to control a compliant robot (F).

We showed good performance of the controller with a simulated compliant quadruped robot on a slope, both on the training test and the testing set. To keep balance on the slope, the robot developed an original strategy consisting in putting one leg more forward than the others, very much unlike what is done in traditional control. The closed-loop controller always performed better than the open-loop gait and reached 85% success on random slopes up to 55%, against 26% for the open-loop gait. Our controller was not as efficient on the randomized height map as on the slope, reaching 57% success against 25% for the open-loop controller. It still performed better in 74% on the cases than the open-loop controller. This relatively lower performance can be explained by the fact that the randomized height map contains significantly more stochasticity than the slope, and the robot may need more complex feedback mechanisms to overcome it. We showed that introducing the phase of the CPG as input to the neural network greatly improved the performance when crossing randomized uneven terrain, compared to when using only the sensor input. This allows the robot to learn phase-dependent feedback mechanisms which bring the success rate to 74%.

One of the strength of our approach is the simplicity of fusing information from different sensors and thus future work should aim at fusing different sensors as in Chapter 5 to further improve the performance. For instance, including information from force sensors on the feet of the robot could greatly help the stabilization in the randomized height map scenario, since the controller would be able to detect missing contacts and learn leg extension and stumbling reflexes. Having phase-dependent feedback could also improve the controller performance, and can be seamlessly implemented by including the phase of one or more oscillators of the CPG as input to the neural network. However, preliminary experiments showed no increase of performance when using the phase of the CPG as an additional neural network input. Finally we are starting to implement our controller on the real Oncilla robot, and comparing it to the model-based feedback controller presented in [6].

Chapter 7

Conclusion

This thesis is coming to an end and while the details of the work achieved have been described in the previous chapters, it is worth summarizing the original contributions of this work to the scientific community. To conclude this thesis I will give a list of these contributions and how I believe the questions originally asked in the introduction of this book were answered. Finally I will discuss the results obtained and what further studies should aim for when addressing similar problems.

7.1 Original Contributions

High level interfacing of vision and CPGs (Chapter 2):
We presented a modular framework to interface vision with a low-level CPG controller for a humanoid robot: the iCub. After the 3D position of markers is extracted from the camera images, a path planning algorithm based on artificial potential fields computes the heading direction for a collision free path to specific goals. An inverse kinematics module computes joint level targets to allow the robot to reach specific positions with its hands. The same inverse kinematics module is used to achieve adaptive drumming with the iCub. The CPG based low-level controller modifies its targets in a smooth way to achieve these high-level goals. While the algorithms used in each of these modules are not novel as such, the original contribution lies here in interfacing all these modules into a controller capable of displaying rich behaviors with CPGs as a base. It is also one of the first display of a non-holonomic legged robot performing navigation in a complex environment using only its embedded cameras.

Fast converging Adaptive Frequency Oscillators (Chapter 3):
We presented a modification of the original Adaptive Frequency Oscillators to allow for faster convergence while conserving a smooth output and the stability

properties of the original AFOs. We used two phases per AFO, one learning the frequency of the forcing signal while the other is only used to generate the output. This allows to set a strong coupling on the phase learning the frequency for fast convergence, and a weak one on the one generating the output in order to avoid the deformation of the output. We showed that this AFO can learn the frequency of the input signal significantly faster than the original version while keeping the output close to a sine wave.

Interfacing vision and AFOs for gaze stabilization (Chapter 3):
We presented an application of the aforementioned modified AFO to the problem of gaze stabilization. By extracting optical flow from the camera images and feeding relevant parts of this flow to three AFOs controlling the three degrees of freedom of a pitch-roll-yaw camera, we managed to stabilize the gaze of the robot. The novelty of this approach is that it does not require a fast sensor like an IMU, but relies only on vision. This is made possible by the phase locking mechanism of the AFOs which effectively generates compensatory commands before the sensor values arrive. Furthermore this approach is the first to investigate AFOs in closed loop both for frequency and amplitude. Indeed in [18], the authors investigate AFOs in closed loop by forcing it with a gyroscope. However, only the frequency is learned here while the amplitude remains constant. In [91] both frequency and amplitude are learned but the output is explicitly subtracted from the input while here the reduction of the optical flow (the input) is a resultant of the camera movements. Interestingly, we in fact learn a different signal than the one the AFO is forced with. We showed applications of the system on different robots in simulation and on real platform as well as a real humans. While the performance of the system was good when the motion of the head was close to a sine, it decreases substantially for non-periodic and step-like movements. This is the main limitation of our system which in theory could be overcome by using multiple AFOs per degrees of freedom, but in practice this meets stability issues.

A unified framework for learning closed loop stability (Chapter 4):
A framework to learn stability for a CPG-controlled legged robot was presented. While most approaches integrate sensory feedback in CPGs by writing explicit equations for the feedback functions (e.g. [57]) or deriving these equations from a kinematic model (e.g. [6]) we chose to represent the mapping between sensors and modifications of the CPG dynamics by a neural network. This neural network should be viewed as a general mapping which weights are learned by particle swarm optimization to achieve stability. We explained how this framework can seamlessly perform sensor fusion and be applied in theory to any robot with any number of sensors and actuators, given sufficient number of neurons, thus sufficient learning time and computational power. This system may also be used in discrete mode (by setting zero amplitude for each oscillator), in rhythmic mode

or a combination of both (meant either as a rhythmic motion with discretely changing offsets or some joints performing rhythmic movements while others only perform discrete ones).

Learning postural control with a compliant humanoid robot (Chapter 5):
We applied the aforementioned framework in discrete mode to the problem of controlling the posture of a compliant humanoid robot, the Coman, standing on a moving platform. This model-free method was also compared systematically to a model-based approach based on a closed form kinematic model. We showed that the model-free approach, when properly trained, performed better than the model-based approach regardless of the frequency of the movements, and the sensor used (gyroscope or optical flow). We also showed that combining vestibular and visual sensors leads to a further increase of the performance of the system, compared to when using each sensor separately. By mapping sensors representing velocities (rotational rates or pixel velocities), to joint speed perturbations (inside the CPG dynamics), the neural network is able to encapsulate information about the robot dynamics, and not only its kinematics. This can explain the better performance of the model-free approach compared to the model-based one. The model-based approach however is more generic than the model-free one since the latter needs in principle to be retrained for each task (different initial posture etc.). The model-free approach was ported without any retuning to the real robot with good performance.

Learning gait stabilization with a compliant quadruped robot (Chapter 6):

We applied the same framework in rhythmic mode this time to the problem of stabilizing an existing open-loop gait stable on flat terrain when the robot has to adapt to non-flat environments, like slopes or randomized height maps. The system displayed good performance when adapting to slopes and height maps, whether using gyro input or vision. The system also showed powerful to ameliorate an open-loop gait on flat ground. The performance on height maps was originally not as good as on slopes however. We introduced the phase of the CPG into the learning framework which greatly improved the performance on random uneven terrain. As for the humanoid postural control problem, this system should show better performance than a pure kinematic-based one (virtual model control etc.), but should also be less generic. The network should in principle be retrained for each task. We investigated training one single network for both slope and height maps environments without success. While the resulting controller reached good performance on slopes, it performed quite badly on height maps. The training time should be increased substantially (possibly exponentially) with the number of different tasks.

7.2 Questions and Answers

Let us review the questions asked in the introduction of this document and how this thesis answers them. Note that I do not intent to give definite solutions for the problems addressed but rather give elements of answers to guide further research.

A *What type of visual information is relevant for which subproblem of locomotion ?*
This question was addressed in all chapters of this thesis. In Chapter 2, we used a marker tracker to obtain precise 3D positioning of object in the scene, thus focusing on the exteroceptive part of vision to achieve the high-level tasks of path planning, reaching and drumming. The work carried out in Chapters 3, 5 and 6 focused more on the exproprioceptive part of vision (here optical flow) since movements of different parts of the robot needed to be estimated (movements of the head in Chapter 3, movements of the whole robot in Chapters 5 and 6).

B *What needs to be modulated in the locomotion patterns to allow adaptation of the nominal gait to rough terrain ?*
In Chapter 4, we explained how to integrate sensory feedback in a CPG so that it modulates the patterns generated to cope with rough environments. We showed how introducing sensory feedback on each of the three variables (radius, phase and offset) of each oscillator allows for rich adaptation of the relevant parameters of the patterns generated (amplitude, frequency, offset and thus local waveform). While the feedback on the phase was not always used in the resulting closed-loop gait, both the feedback on the offset and radius were used extensively. Thus we directly addressed this question, and further applications and results illustrating this method were given in Chapters 5 and 6.

C *How to interface the visual system with the locomotor system ?*
This question is another central one which was addressed throughout this thesis. The most direct way of interfacing vision with locomotion was described in Chapter 3 where the AFOs controlling the head movements were directly entrained by the raw optical flow. This creates a direct link between layers 2 (*Sensor Processing*) and layer 5 (*Low Level Control*) of Figure 1.3 presented in Section 1.5.3 of the introduction of this thesis (link 7 in the figure). In Chapters 4, 5 and 6, we showed how to learn a proper mapping of vision with the locomotion controller. While the learning process is rather computationally demanding, once learned, the whole feedback controller consists in a simple feedforward neural network thus a simple series of sigmoid computations from the raw optical flow to the CPG modulations. In Chapter 5 we also showed how the absolute orientation of the robot can be estimated from visual information (a computation which should be viewed as part of the *High Level Control* layer), and passed down to the lower layers (kinematic model then low-level joint controller). Finally in Chapter 2 we interfaced vision with the low-level controller by going through all layers of the architecture. Images were processed by the *Sensor Processing* layer (2) to extract marker positions which were passed on to the *High Level Control* layer (3). This marker positions were translated into the frame of the

robot and a path was computed by the *High Level Control* layer and then passed on to the *Mid Level Control*, in charge of translating these high-level goals into targets at the joint level for the *Low Level Controller*. Using several layers between the sensors and low-level controller has the advantage of breaking down the complex problem of closed-loop locomotion into smaller and in principle easier subproblems. However it comes at the cost of additional delays between these two layers. This is not an issue for problems like path planning where the robot is anyway anticipating its motion at least a few seconds in advance, but can be critical for balance control for instance where any delay may cause the controller to become unstable. Furthermore, the summed complexity of the computation performed in each layer is not necessarily less than that of more direct mappings. For head stabilization for instance, usual controllers can be decomposed into four steps: head state estimation, head posture adjustments, error computation and controller gains adjustments. Each of these steps constitutes a relatively complex and high dimensional problem, while a system like ours bypasses most of these issues.

D *How to fuse information from the visual system and the other sensors ?*
While sensor fusion is not the main subject of any Chapter of this thesis, the balance controller described in Chapter 4 was designed such that it could perform sensor fusion in a simple way. Fusing vision with gyroscope values (vestibular information) was addressed in Chapters 5 and 6 to perform postural control and gait stabilization. We performed sensor fusion simply by feeding different sensor values to the neural network and showed that this increased the performance of our system. This can be explained mainly by the fact that it introduces redundancy in the sensor values, and thus minimizes the effect of noise on each single sensor. However, our system is not (yet) capable of adapting the importance of each sensor online. In contrast the animal brain performs constant re-weighting of the sensor inputs at the neural level, to give more emphasis on sensors with higher accuracy. Extreme examples include blind people who rely solely on their inner ear to keep balance, while people with vestibular disorders cannot afford to close their eyes when walking or even standing. The next generation of out balancing system should include re-weighting of the sensory cues, which would increase the interest of fusing different sensors.

E *How can locomotion be modulated to enhance the visual capabilities of a robot ?*
This question is the main topic of Chapter 3 which is also the only topic where it is addressed. We presented a system capable of stabilizing the gaze of a robot (or a pan-tilt camera) when performing periodic locomotion or when tracking a periodically moving object. By stabilizing the camera, the scene or object is maintained steady in the field of view, causing motion blur to be reduced and rendering further image processing much more efficient.

F *How can vision help controlling a compliant system ?*
This is also a central question for Chapters 5 and 6. Both chapters aim at controlling compliant robots: a compliant humanoid robot for Chapter 5 and a compliant quadruped robot for 6. While these two robots are very different in morphology, mass,

actuation mechanism etc. we control both of them with the same controller with only slight modifications. In Chapter 5 we show that a pure kinematic controller quickly shows its limits for controlling the posture of a compliant robot. Neglecting the effect of the springs seem to lead to instabilities of the control, by self-excitation of the springs and the controller. In contrast, the controller presented here includes these effects in its learning process, leading to better performance.

7.3 Discussion and Future Work

The work of this thesis has given insights into how to use vision at different levels of the control of legged robots. We gave an example of a fully integrated framework where vision is used to define high-level goals which are then passed on to the lower layers. We also showed how to use optical flow as a more direct sensory feedback to the low-level control, with applications for gaze stabilization, postural control and gait stabilization on rough terrain.

While this work investigated uses of vision seldom addressed in the literature, I strongly believe vision can bring even more to the field of robot locomotion control. First of all, in this thesis, we only used exproprioceptive visual information for controlling stabilizing the gait of the robot (Chapters 5 and 6) when walking on rough terrain. Thus, the robot only adapts its gait whenever perturbations from the environment occur (whenever it starts falling). In theory, using our system, the robot could extract information about the environment ahead from the optical flow and use it to modulate its gait in advance. However, The magnitude of the optical flow due to the self-motion of the robot is much larger than the deflection due to the shape of the environment, so the robot most likely learns only to compensate from its self-motion. One interesting axis of research for future studies should consist in including extroceptive information in the stabilization process (the full depth map of the environment ahead for instance). The robot would be able to modulate its locomotion patterns before the rough environment actually arrives to improve stability, place its feet at specific "safe" positions or avoid difficult sections.

With 3D visual sensors getting more and more widespread, it would be interesting to try our approach with 3D optical flow as input. In three dimensions, some motions are easier to discriminate (pitching rotation and vertical translation for instance). Both the gaze stabilization system presented in Chapter 3 and the balance control framework described in Chapters 4, 5 and 6 might benefit from 3D optical flow information. In addition, a more elaborate analysis of the optical flow could benefit the controller. We could for instance analyze the focus of expansion, or perform a statistical dimensionality reduction of the optical flow (e.g. Principal Component Analysis) instead of a geometrical one, before providing it to the neural network. A convolution neural network could also be used as a first layer of the neural-network to do automatic feature extraction.

We chose to use a feedforward neural-network as sensorimotor mapping for the control of balance in Chapters 4, 5 and 6. While this model is simple to train and analyze, it constitutes only a static mapping. Sensor values arriving at time t only influence the outputs of the network at time t. However, compliance in robots introduces additional delays and more complex dynamics (for a series elastic joint, energy transferred from the motor to the spring at a certain time is only transferred from the spring to the robot body a moment later). More complex structure of the neural model could be imagined. For instance one could try using time delay neural networks, leaky integrator networks, recurrent neural networks, or even reservoir networks. This would allow the system to learn proper delays and memory effects between the sensors and the CPG modulations. We did preliminary experiments with reservoir networks since they have the added advantage of only needing to tune the output weights of the network, thus decreasing the search space for particle swarm optimization. However, these experiments seem to suggest that, due to the very high non-linearity of the reservoir, the fitness landscape represented by the output weights become very rough, which leads to PSO failing to find good solutions. Reservoir regularization techniques exist that may solve this problem. In addition to encapsulating memory, these more complex neural models could also increase the amount of information stored. While our feedforward neural network generalized well in between situations it has been trained for, it should be re-trained for very different situations (ex. trained once for slopes and once for rocky terrain). A more complex neural model could allow to train one single network for different environments, although the training would most likely become more complex and computationally demanding.

In this thesis we controlled all robots in position. While this makes the control problem simpler, the general trend is to move towards torque or force control for rough terrain locomotion. This allows for active and adaptive compliance of the robot. It would be interesting to investigate whether our system can be used to modulate torque patterns instead of position patterns.

We mainly considered here vestibular and visual sensors. More sensors like torque/load sensors, touch sensors or encoders after the spring could bring precious information to the stabilization controller. For gaze stabilization, we could also try using a gyroscope or accelerometer as input to the AFO and compare it to visual input.

An interesting extension to the robot stabilization approach presented here could consist in adding a second task on top of the balance control. For instance one could try to have the robot reach different positions with its hands while the platform is moving. One approach to handling that would be to simply train our controller with different frequencies for the platform motion but also with different reaching targets. The secondary task (the reaching motion) would then be considered simply as an additional perturbation in the sensor space, which the neural network would learn to compensate for.

Finally the main limitation of our balance controller is that it relies on a good simulated model and offline learning. Further studies should aim at designing controller which can learn the modulations of the gait patterns online, directly on the real robot. Online learning is very widespread for manipulators, wheeled robots and flying robots, but legged robots have a major drawback in that regard: they can fall. When a manipulator fails to achieve the given task, let's say cup and ball, it can detect that and simply try again. When a legged robot falls on rough terrain some procedure needs to be designed to bring it back to its initial position and continue the learning process (assuming it did not break). This can be both time consuming and complex in terms of mechanical design. Although we had good success transferring the controller evolved in simulation to the real robot (see Chapter 5, Section 5.6), I believe designing some method to at least continue learning online (possibly throughout the life of the robot), after a first learning phase in simulation, should be given top priority for further research.

Appendix A

Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a population-based stochastic optimization method inspired by the movements of individuals in a swarm (of fish, birds etc.). It was developed in 1995 by R. Eberhart and J. Kennedy ([54]). PSO is used to maximize a fitness function (or minimize a cost function) in a multi-dimensional search space where the fitness landscape is non convex. It shares many similarities with Genetic Algorithms, also population-based and bio-inspired, the main difference being that Genetic Algorithms are based on competition between individuals to "survive", while PSO is based on collaboration between them.

As all population-based methods, PSO works by generating a randomly distributed initial population, where individuals define values of a set of parameters to optimize, then evaluating the performance of this population on the task to solve and iteratively improving it. However, unlike Genetic Algorithms which select the few best individuals and generates the next population by "crossing" them, PSO keeps all individuals (particles) throughout the course of the optimization. These particles never "die" but move around the search space, ideally until they all meet in one point which may be the global maximum or just a local one. At each iteration, PSO updates the velocity of each particle according to three criteria:

- its current velocity
- its own best position so far (with maximum fitness or minimum cost)
- the best position of all individuals so far.

Thus, as in animal swarms, individuals communicate with each other to update their trajectory.

Next we detail the different steps of the algorithm.

Initialization: Initialize a uniformly distributed initial population of N individuals in an M-dimensional search space (N is to be chosen by the user according to the

number of parameters to optimize). Both position and velocity of each particle i are uniformly distributed.

$$x_{i,0} \sim \mathcal{U}(b_-, b_+) \tag{A.1}$$
$$v_{i,0} \sim \mathcal{U}(-|b_+ - b_-|, |b_+ - b_-|) \tag{A.2}$$

where $x_{i,0}$ is the position of the i_{th} particle at iteration 0, $v_{i,0}$ its velocity and b_- and b_+ are M-dimensional lower and upper bounds of the search space chosen by the user.
Note that while the uniform distribution is usually chosen when no assumption can be made on the location of the best set of parameters, other priors may be better suited when the expert can make an educated guess. Variants exist also on the velocity initialization, for instance by starting with zero velocities.

Iterations: For a fixed number of iterations or until a convergence criterion is met, update the particles velocity and then position according to (for the k^{th} iteration):

$$v_{i,k+1} = \omega v_{i,k} + \phi_p r_p (X_{i,k} - x_{i,k}) + \phi_g r_g (X_{g,k} - x_{i,k}) \tag{A.3}$$
$$x_{i,k+1} = x_{i,k} + v_{i,k} \tag{A.4}$$

where r_p and r_g are random numbers ($rp, rg \sim \mathcal{U}(0,1)$). ω (the *inertia*), ϕ_p (the *cognitive factor*) and ϕ_g (the *social factor*) are scaling factors chosen by the user. These parameters define the trade-off between exploration and exploitation, and greatly influence the performance of PSO on particular problems. $X_{i,k}$ is the best known position of the particle i until iteration k, while $X_{g,k}$ is the best know position in the whole swarm.
Each particle is then evaluated on the task, $X_{i,k}$ and $X_{g,k}$ are updated if necessary and the update rules on the position and velocity of each particles are repeated until the termination criterion is met.

Some other parameters can be set in PSO like the behavior of particles reaching the boundary of the search space (bouncing, sticking etc.), or the maximum velocity of particles.
For this work we used $\phi_p = \phi_g = 2.05$ and $\omega = 0.729$, which are widely used default values.

Bibliography

[1] Video: Learning robot gait stability. *http://www.youtube.com/watch?v=ef_Z6RUc_6g*.

[2] Video: Model-based and model-free approaches for postural control of a compliant humanoid robot. *http://www.youtube.com/watch?v=TH-9ZrmGQY0*.

[3] Video: Path planning and reaching with icub. *http://www.youtube.com/watch?v=GWG8RVWL_fo*.

[4] Video: Robot gaze stabilization during locomotion using adaptive frequency oscillators. *http://www.youtube.com/watch?v=GWG8RVWL_fo*.

[5] J. K. Aggarwal and N. Nandhakumar. On the computation of motion from sequences of images - a review. *P Ieee*, 76(8):917–935, Jan 1988.

[6] M. Ajallooeian, S. Gay, A. Tuleu, A. Sproewitz, and A. J. Ijspeert. Modular Control of Limit Cycle Locomotion over Unperceived Rough Terrain. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013.

[7] M. Ajallooeian, S. Pouya, A. Sproewitz, and A. Ijspeert. Central pattern generators augmented with virtual model control for quadruped rough terrain locomotion. In *IEEE International Conference on Robotics and Automation (ICRA 2013)*, 2013.

[8] M. a. Ali, H. a. Park, and C. S. G. Lee. Closed-form inverse kinematic joint solution for humanoid robots. *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 704–709, Oct. 2010.

[9] AMARSI. http://amarsi.soltoggio.net/. Adaptive Modular Architectures for Rich Motor Skills (temporary website).

[10] D. E. Angelaki, Y. Gu, and G. C. DeAngelis. Multisensory integration: psychophysics, neurophysiology, and computation. *Curr Opin Neurobiol*, 19(4):452–458, Jan 2009.

[11] V. Barasuol, J. Buchli, C. Semini, M. Frigerio, E. De Pieri, and D. Caldwell. A Reactive Controller Framework for Quadrupedal Locomotion on Challenging Terrain. *Accepted for IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[12] B. G. Bardy, W. H. Warren, and B. a. Kay. The role of central and peripheral vision in postural control during walking. *Perception & psychophysics*, 61(7):1356–68, Oct. 1999.

[13] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical-flow techniques. *Int J Comput Vision*, 12(1):43–77, Jan 1994.

[14] D. Bernardin, H. Kadone, D. Bennequin, T. Sugar, M. Zaoui, and A. Berthoz. Gaze anticipation during human locomotion. *Experimental brain research. Experimentelle Hirnforschung. Expérimentation cérébrale*, 223(1):65–78, Nov. 2012.

[15] S. Bertrand, O. Bruneau, F. Ouezdou, and S. Alfayad. Closed-form solutions of inverse kinematic models for the control of a biped robot with 8 active degrees of freedom per leg. *Mechanism and Machine Theory*, 49:117–140, Mar. 2012.

[16] BioRob. http://biorob.epfl.ch/. BioRobotics Laboratory at EPFL.

[17] J. Borenstein, Y. Koren, and S. Member. The vector field histogram - fast obstacle avoidance for mobile robots. *IEEE Journal of Robotics and Automation*, 7:278–288, 1991.

[18] J. Buchli, F. Iida, and A. Ijspeert. Finding Resonance: Adaptive Frequency Oscillators for Dynamic Legged Locomotion. *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3903–3909, Oct. 2006.

[19] J. Buchli and A. Ijspeert. Self-organized adaptive legged locomotion in a compliant quadruped robot. *Autonomous Robots*, pages 331–347, 2008.

[20] J. Buchli, M. Kalakrishnan, M. Mistry, P. Pastor, and S. Schaal. Compliant quadruped locomotion over rough terrain. *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 814–820, Oct. 2009.

[21] J. Chang, W. Hu, M. Cheng, and BS. Digital Image Traslational and Rotational Motion Stabilization Using Optical Flow Technique. In *IEEE Transactions on Consumer Electronics*, volume 48, 2002.

[22] J. Chestnutt, M. Lau, K. M. Cheung, J. Kuffner, J. K. Hodgins, and T. Kanade. Footstep planning for the honda asimo humanoid. In *Proceedings of the IEEE International Conference on Robotics and Automation*, April 2005.

[23] M. E. Cinelli, A. E. Patla, and F. Allard. Behaviour and gaze analyses during a goal-directed locomotor task. *Quarterly journal of experimental psychology (2006)*, 62(3):483–99, Mar. 2009.

[24] A. H. Cohen. Adaptive Dynamic Walking of a Quadruped Robot on Natural Ground Based on Biological Concepts. *The International Journal of Robotics Research*, 2007.

[25] P. Dean, J. Porrill, and J. V. Stone. Decorrelation control by the cerebellum achieves oculomotor plant compensation in simulated vestibulo-ocular reflex. *Proceedings. Biological sciences / The Royal Society*, 269(1503):1895–904, Sept. 2002.

[26] S. Degallier, L. Righetti, S. Gay, and A. Ijspeert. Towards autonomous robotic control based on discrete and rhythmic motor primitives. *In preparation for Autonomous Robots*, 2010.

[27] S. Degallier, L. Righetti, S. Gay, and A. Ijspeert. Toward simple control for complex, autonomous robotic applications: combining discrete and rhythmic motor primitives. *Autonomous Robots*, 31:155–181, 2011.

[28] S. Degallier, L. Righetti, L. Natale, F. Nori, G. Metta, and A. Ijspeert. A modular bio-inspired architecture for movement generation for the infant-like robot iCub. In *Proceedings of the 2nd IEEE RAS / EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, 2008.

[29] A. P. Duchon and W. H. Warren. A Visual Equalization Strategy for Locomotor Control: Of Honeybees, Robots, and Humans. *Psychological Science*, 13(3):272–278, May 2002.

[30] M. Egelhaaf and R. Kern. Vision in flying insects. *Current Opinion in Neurobiology*, 12(6):699–706, Dec. 2002.

[31] D. J. Fleet and A. D. Jepson. Computation of component image velocity from local phase information. *Int. J. Comput. Vision*, 5(1):77–104, 1990.

[32] H. Forssberg. Stumbling corrective reaction: a phase-dependent compensatory reaction during locomotion. *Journal of neurophysiology*, 42(4):936–53, July 1979.

[33] H. Forssberg, S. Grillner, and S. Rossignol. Phase dependent reflex reversal during walking in chronic spinal cats. *Brain research*, 85(1):103–7, Feb. 1975.

[34] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, 4, 1997.

[35] N. Franceschini, F. Ruffier, and J. Serres. A bio-inspired flying robot sheds light on insect piloting abilities. *Current biology : CB*, 17(4):329–35, Feb. 2007.

[36] Y. Fukuoka, T. Mimura, N. Yasuda, and H. Kimura. Integration of multi sensors for adaptive walking of a quadruped robot. In *Multisensor Fusion and Integration for Intelligent Systems, MFI2003. Proceedings of IEEE International Conference on*, pages 21 – 26, 30 2003.

[37] S. Gay, S. Degallier, U. Pattacini, and A. Ijspeert. Integration of vision and central pattern generator based locomotion for path planning of a nonholonomic crawling humanoid robot. *Submitted to the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010)*, 2010.

[38] S. Gay, A. Ijspeert, and J. Santos-Victor. Robot Head Stabilization During Periodic Locomotion Using Adaptive Dynamical Systems. *5th International Symposium on Adaptive Motion of Animals and Machines (AMAM2011)*, pages 43–44, 2011.

[39] S. Gay, A. Ijspeert, and J. Santos Victor. Predictive gaze stabilization during periodic locomotion based on Adaptive Frequency Oscillators. *2012 IEEE International Conference on Robotics and Automation*, pages 271–278, May 2012.

[40] S. Gay, J. Santos-Victor, and A. Ijspeert. Learning Robot Gait Stability using Neural Networks as Sensory Feedback Function for Central Pattern Generators. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013.

[41] S. Gay, J. Santos-Victor, and A. J. Ijspeert. Model-Based and Model-Free Approaches for Postural Control of a Compliant Humanoid Robot using Optical Flow. In *IEEE-RAS International Conference on Humanoid Robots*, 2013.

[42] R. Geraerts and M. H. Overmars. Sampling techniques for probabilistic roadmap planners. *International Conference on Intelligent Autonomous Systems*, 2004.

[43] M. Gerin-Lajoie, C. L. Richards, J. Fung, and B. J. McFadyen. Characteristics of personal space during obstacle circumvention in physical and virtual environments. *Gait & posture*, 27(2):239–247.

[44] A. Goswami. Postural stability of biped robots and the foot-rotation indicator (FRI) point. *The International Journal of Robotics Research*, 1999.

[45] H. Hauser, G. Neumann, A. J. Ijspeert, and W. Maass. Biologically inspired kinematic synergies enable linear balance control of a humanoid robot. *Biological cybernetics*, 104(4-5):235–49, May 2011.

[46] T. Higuchi. Visuomotor Control of Human Adaptive Locomotion: Understanding the Anticipatory Nature. *Frontiers in Psychology*, 4(May):1–9, 2013.

[47] T. Higuchi, M. E. Cinelli, M. a. Greig, and A. E. Patla. Locomotion through apertures when wider space for locomotion is necessary: adaptation to artificially altered bodily states. *Experimental brain research. Experimentelle Hirnforschung. Expérimentation cérébrale*, 175(1):50–9, Oct. 2006.

[48] A. J. Ijspeert. Central pattern generators for locomotion control in animals and robots: a review. *Neural Networks*, 21(4):642–653, 2008.

[49] A. J. Ijspeert, A. Crespi, D. Ryczko, and J.-M. Cabelguen. From swimming to walking with a salamander robot driven by a spinal cord model. *Science*, 315(5817):1416–1420, 2007.

[50] S. Inagaki, H. Yuasa, T. Suzuki, and T. Arai. Wave cpg model for autonomous decentralized multi-legged robot: Gait generation and walking speed control. *Robotics and Autonomous Systems*, 54(2):118 – 126, 2006. Intelligent Autonomous Systems.

[51] D. Jo and K. Didier. A reactive robot navigation system based on a fluid dynamics metaphor. In *Parallel Problem Solving from Nature*, volume 496 of *Lecture Notes in Computer Science*, pages 355–362. Springer Berlin / Heidelberg, 1991.

[52] S. Kagami, K. Nishiwaki, J. Kuffner, J.J., Y. Kuniyoshi, M. Inaba, and H. Inoue. Online 3d vision, motion planning and bipedal locomotion control coupling system of humanoid robot: H7. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, pages 2557 – 2562 vol.3, 2002.

[53] M. Kawato. Feedback-error-learning neural network for supervised motor learning. *Advanced neural computers*, 6(3):365–372, 1990.

[54] J. Kennedy. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948. Ieee, 1995.

[55] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. Rob. Res.*, 5(1):90–98, 1986.

[56] H. Kimura, S. Akiyama, and K. Sakurama. Realization of dynamic walking and running of the quadruped using neural oscillator. *Auton. Robots*, 7(3):247–258, 1999.

[57] H. Kimura, S. Akiyama, and K. Sakurama. Realization of Dynamic Walking and Running of the Quadruped Using Neural Oscillator. *Autonomous Robots*, 258:247–258, 1999.

[58] J. Z. Kolter and A. Y. Ng. Stereo vision and terrain modeling for quadruped robots. *2009 IEEE International Conference on Robotics and Automation*, pages 1557–1564, May 2009.

[59] J. Z. Kolter and A. Y. Ng. The Stanford LittleDog : A Learning and Rapid Replanning Approach to Quadruped Locomotion. pages 1–48, 2010.

[60] D. Kragic and M. Vincze. Vision for Robotics. *Foundations and Trends in Robotics*, 1(1):1–78, 2009.

[61] S. Lacroix, A. Mallet, D. Bonnafous, G. Bauzil, S. Fleury, M. Herrb, and R. Chatila. Autonomous rover navigation on unknown terrains: Functions and integration. *The International Journal of Robotics Research*, 21(10-11):917–942, October 2002.

[62] D. N. Lee and D. S. Young. Gearing action to the environment. *Experimental Brain Research*, SUPPL. 15:217–230, 1986. cited By (since 1996) 12.

[63] A. Lenz, T. Balakrishnan, a. G. Pipe, and C. Melhuish. An adaptive gaze stabilization controller inspired by the vestibulo-ocular reflex. *Bioinspiration & Biomimetics*, 3(3):035001, Sept. 2008.

[64] M. Lewis. Perception driven robot locomotion. *Journal Robot Society of Japan*, 20(April):51–56, 2002.

[65] M. A. Lewis and L. S. Simó. Elegant stepping: A model of visually triggered gait adaptation. *Connection Science, Special Issue on Adaptive Robotics, Vol 11*, Jan 1999.

[66] J. R. Lishman and D. N. Lee. The autonomy of visual kinaesthesis. *Perception*, 2(3):287–294, 1973.

[67] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *International joint conference on artificial intelligence*, number x, pages 674–679, 1981.

[68] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI'81: Proceedings of the 7th international joint conference on Artificial intelligence*, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[69] C. Maufroy, H. Kimura, and K. Takase. Towards a general neural controller for quadrupedal locomotion. *Neural networks : the official journal of the International Neural Network Society*, 21(4):667–81, May 2008.

[70] O. Michel. Webots: Professional mobile robot simulation. *Journal of Advanced Robotics Systems*, 1(1):39–42, 2004.

[71] B. J. Mohler, W. B. Thompson, S. H. C.-r. Herbert, L. P. Jr, and W. H. W. Jr. Visual flow influences gait transition speed and preferred walking speed. *Experimental Brain Research*, pages 221–228, 2007.

[72] R. Moraes, M. A. Lewis, and A. E. Patla. Strategies and determinants for selection of alternate foot placement during human locomotion: influence of spatial and temporal constraints. *Experimental brain research. Experimentelle Hirnforschung. Expérimentation cérébrale*, 159(1):1–13, Nov. 2004.

[73] M. L. Morgan, G. C. DeAngelis, and D. E. Angelaki. Multisensory integration in macaque visual cortex depends on cue reliability. *Neuron*, 59(4):662–673, Jan 2008.

[74] M. Mossio, M. Vidal, and A. Berthoz. Traveled distances: new insights into the role of optic flow. *Vision research*, 48(2):289–303, Jan. 2008.

[75] M. P. Murphy, a. Saunders, C. Moreira, a. a. Rizzi, and M. Raibert. The LittleDog robot. *The International Journal of Robotics Research*, 30(2):145–149, Dec. 2010.

[76] N. Oda, J. Yoneda, and T. Abe. Visual feedback control of ZMP for biped walking robot. *IECON 2011 - 37th Annual Conference of the IEEE Industrial Electronics Society*, pages 4543–4548, Nov. 2011.

[77] C. Ott, M. a. Roa, and G. Hirzinger. Posture and balance control for biped robots based on contact force optimization. *2011 11th IEEE-RAS International Conference on Humanoid Robots*, pages 26–33, Oct. 2011.

[78] D. Owaki, T. Kano, K. Nagasawa, A. Tero, and A. Ishiguro. Simple robot suggests physical interlimb communication is essential for quadruped walking. *Journal of the Royal Society, Interface / the Royal Society*, (October), Oct. 2012.

[79] F. Panerai, G. Metta, and G. Sandini. Learning visual stabilization reflexes in robots with moving eyes. *Neurocomputing*, 48(1-4):323–337, Oct. 2002.

[80] S. Park, Y. Han, and H. Hahn. Balance control of a biped robot using camera image of reference object. *International Journal of Control, Automation and Systems*, 7(1):75–84, Mar. 2009.

[81] A. Patla. How is human gait controlled by vision. *Ecological Psychology*, Jan 1998.

[82] A. Patla. How is human gait controlled by vision. *Ecological Psychology*, (786945360), 1998.

[83] A. E. Patla. Understanding the roles of vision in the control of human locomotion. *Gait & Posture*, 5(1):54–69, Feb. 1997.

[84] D. Pongas, M. Mistry, and S. Schaal. A Robust Quadruped Walking Gait for Traversing Rough Terrain. *Robotics and Automation, 2007 ...*, 2:1–6, 2007.

[85] J. Pratt, C. Chew, and A. Torres. Virtual model control: An intuitive approach for bipedal locomotion. *International Journal of Robotics Research*, (129), 2001.

[86] M. Raibert. Legged Robots that Balance. 1985.

[87] M. Raibert and K. Blankespoor. Bigdog, the roughterrain quadruped robot. *Proceedings of the 17th ...*, pages 6–9, 2008.

[88] M. Raibert, M. Chepponis, and H. Brown. Running on four legs as though they were one. *Robotics and Automation, IEEE Journal of*, 2(2):70–82, 1986.

[89] M. Raibert and J. Hodgins. Legged robots. In R. Beer, R. Ritzmann, and T. McKenna, editors, *Biological Neural Networks in Invertebrate Neuroethology and Robotics*, pages 319–354. Academic Press, 1993.

[90] F. Raudies, E. Mingolla, and H. Neumann. Active gaze control improves optic flow-based segmentation and steering. *PloS one*, 7(6):e38446, Jan. 2012.

[91] L. Righetti, J. Buchli, and A. Ijspeert. Dynamic Hebbian learning in adaptive frequency oscillators. *Physica D: Nonlinear Phenomena*, 216(2):269–281, Apr. 2006.

[92] L. Righetti and A. Ijspeert. Programmable central pattern generators: an application to biped locomotion control. In *2006 IEEE International Conference on Robotics and Automation (ICRA 2006)*, number May, pages 1585–1590. IEEE, 2006.

[93] L. Righetti and A. J. Ijspeert. Pattern generators with sensory feedback for the control of quadruped locomotion. In *Proceedings of the 2008 IEEE International Conference on Robotics and Automation (ICRA 2008)*, pages 819–824, 2008.

[94] RobotCub. http://www.robotcub.org/. An Open Framework for Research in Embodied Cognition.

[95] C. P. Santos, M. Oliveira, A. M. a. C. Rocha, and L. Costa. Head motion stabilization during quadruped robot locomotion: Combining dynamical systems and a genetic algorithm. In *2009 IEEE International Conference on Robotics and Automation*, pages 2294–2299. Ieee, May 2009.

[96] K. Sato. Collision avoidance in multi-dimensional space using laplace potential. *in Proc. 15th Conf. Robotics Soc. Jpn.*, 1987.

[97] C. Semini, N. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. Caldwell. Design of hyq a hydraulically and electrically actuated quadruped robot. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 225(6):831–849, 2011.

[98] J. Serres, D. Dray, F. Ruffier, and N. Franceschini. A vision-based autopilot for a miniature air vehicle: joint speed control and lateral obstacle avoidance. *Autonomous Robots*, 25(1-2):103–122, Dec. 2007.

[99] J. Shi and C. Tomasi. Good features to track. In *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 593 – 600, 1994.

[100] J. Shi and C. Tomasi. Good Features to Track. In *Conference on Computer Vision and Pattern Recognition*, number June, 1994.

[101] T. Shibata and S. Schaal. Biomimetic gaze stabilization based on feedback-error-learning with nonparametric regression networks. *Science And Technology*, 14:201–216, 2001.

[102] A. Sproewitz, L. Kuechler, A. Tuleu, M. Ajallooeian, M. D'Haene, R. Moeckel, and A. J. Ijspeert. Oncilla robot: a light-weight bio-inspired quadruped robot for fast locomotion in rough terrain. In *Adaptive Motion in Animals and Machines, 5th International symposium, Abstracts*, 2011.

[103] A. Sproewitz, A. Tuleu, M. Vespignani, M. Ajallooeian, E. Badri, and A. Ijspeert. Towards dynamic trot gait locomotion – design, control, and experiments with a compliant quadruped robot. *Interational Journal of Robotics Research (IJRR)*, 2012, Submitted.

[104] M. Srinivasan and R. Moore. From biology to engineering: Insect vision and applications to robotics. *Frontiers in Sensing*, 2012.

[105] M. V. Srinivasan. An image-interpolation technique for the computation of optic flow and egomotion. *Biological Cybernetics*, 71(5):401–415, September 1994.

[106] M. V. Srinivasan. Visual control of navigation in insects and its relevance for robotics. *Current opinion in neurobiology*, 21(4):535–43, Aug. 2011.

[107] M. V. Srinivasan, M. Poteser, and K. Kral. Motion detection in insect orientation and navigation. *Vision research*, 39(16):2749–66, Aug. 1999.

[108] G. Taga, Y. Yamaguchi, and H. Shimizu. Self-organized control of bipedal locomotion by neural oscillators in unpredictable environment. *Biological Cybernetics*, 65(3):147–159, July 1991.

[109] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney. Winning the darpa grand challenge. *Journal of Field Robotics*, 2006. accepted for publication.

[110] S. Viollet and N. Franceschini. A high speed gaze control system based on the vestibulo-ocular reflex. *Robotics and Autonomous Systems*, 50:147–161, 2005.

[111] R. J. Vogelstein, R. Etienne-Cummings, N. V. Thakor, and A. H. Cohen. Phase-dependent effects of spinal cord stimulation on locomotor activity. *IEEE transactions on neural systems and rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society*, 14(3):257–65, Sept. 2006.

[112] M. Vukobratović and J. Stepanenko. On the stability of anthropomorphic systems. *Mathematical Biosciences*, 37:1–37, 1972.

[113] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106:25–57, 2006.

[114] D. Wagner. http://studierstube.icg.tu-graz.ac.at/handheld_ar/artoolkitplus.php. ARToolkitPlus - Augmented Reality Tracking Library.

[115] W. H. Warren, B. a. Kay, W. D. Zosh, a. P. Duchon, and S. Sahuc. Optic flow is used to control human walking. *Nature neuroscience*, 4(2):213–6, Feb. 2001.

[116] Webots. http://www.cyberbotics.com. Commercial Mobile Robot Simulation Software.

[117] A. Wedel and D. Cremers. *Stereo Scene Flow for 3D Motion Analysis*. Springer London, London, 2011.

[118] D. Wettergreen, H. Thomas, and C. Thorpe. Planning strategies for the ambler walking robot. In *Proceedings of the 1990 IEEE International Conference on Systems Engineering*, pages 198–203, 1990.

[119] J. F. Yang and R. B. Stein. Phase-dependent reflex reversal in human leg muscles during walking. *Journal of neurophysiology*, 63(5):1109–17, May 1990.

[120] J.-c. Zufferey, A. Beyeler, and D. Floreano. Optic Flow to Steer and Avoid Collisions in 3D. In D. Floreano, J.-C. Zufferey, M. V. Srinivasan, and C. Ellington, editors, *Flying Insects and Robots*, pages 73–86. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[121] J.-C. Zufferey and D. Floreano. Fly-inspired visual steering of an ultralight indoor aircraft. *IEEE Transactions on Robotics*, 22(1):137–146, Feb. 2006.

Curriculum Vitae

Sébastien Gay
Chemin du Bré 6, 1023 Crissier,
Switzerland
+41 (0) 78 923 61 07
sebastien.gay@epfl.ch
sebastien.gay.pro@gmail.com
French nationality
Web page:
http://biorob.epfl.ch/people/gay
LinkedIn profile: http://lnkd.in/hER_Fz

Education

PhD in robotics (2009 - 2013): Co-supervision Biorobotics Laboratory at Ecole Polytechnique Fédérale de Lausanne (EPFL) and vision and robotics laboratory at Instituo Superio Técnico (IST), Lisbon.)

***Research* Master in Computer Science (2006-2007):** National Institute of Applied Sciences (INSA), Lyon, Artificial Intelligence major, with honors, valedictorian.

Engineer's diploma (2002-2007): National Institute of Applied Sciences (INSA), Lyon)

Scientific Baccalaureate (2002) *European* Baccalaureate with German specialty, with honors.

Experience

Research assistant
Biorob (EPFL) and Vislab (IST)
2009 - Present
Lausanne, Switzerland and Lisbon, Portugal

Works on using vision to increase the walking performance of legged robots on rough uneven terrain. Develops control software mostly for humanoids and quadrupeds for path planning, camera stabilization, postural control and gait stabilization. Develops a framework allowing a robot to learn by itself to keep balance. This work lead to several publications in international conferences and journals (full list available at http://biorob.epfl.ch/people/gay). Gives a Master's course and supervises practicals and students projects.

robotics | artificial intelligence | image processing | kinematic modeling | C | C++ | C# | Matlab

End of study project (Engineer and Master)
Biorob (EPFL)
April to September 2009
Lausanne, Switzerland

Development of algorithms for reconfiguration planning of modular robots.
Design of a 3D user interface to allow inexperienced users to design modular robots.
See http://biorob2.epfl.ch/pages/studproj/?id=birg66475.
This work served as basis for a PhD project on modular robots and for the user interface currently used.

graph theory | mysql | heuristics | user interface | OpenGL | C++ | Matlab

R&D internship
Sino-French Laboratory for Computer Science, Automation and Applied Mathematics (LIAMA), Chinese Academy of Sciences
April to September 2006
Beijing, China

Development of algorithms for simplification of 3D visualization of conifer foliage, according to the viewpoint in scenes with many objects (mipmapping). Re-design of the group's software framework in C++ to make it modular and object oriented. This work lead to a publication in an international journal.
See
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.101.3127

3D programming | OpenGL | mipmapping | object oriented design | C++ | C

Software engineer internship
ASI
May to August 2005
Lyon, France

Design and reorganization of data bases, business intelligence, data mining and Java programming. This work allowed to enhance the accessibility and visualizing of a company data base.

data base modeling | business intelligence | data mining | SQL | Java

List of Publications

Sébastien Gay, Jesse van den Kieboom, José Santos-Victor and Auke Ijspeert, Model-Based and Model-Free Approaches for Postural Control of a Compliant Humanoid Robot using Optical Flow, *IEEE-RAS International Conference on Humanoid Robots (Humanoids), 2013*, Atlanta, USA, October 15-17 2013

Sébastien Gay, José Santos-Victor and Auke Ijspeert, Learning Robot Gait Stability using Neural Networks as Sensory Feedback Function for Central Pattern Generators, *IEEE International Conference on Intelligent Robots and Systems(IROS), 2013*, Tokyo, Japan, November 3-7 2013

Mostafa Ajallooeian, Soha Pouya, Sébastien Gay, Alexandre Tuleu, Alexander Sprowitz and Auke Ijspeert, Towards Modular Control for Moderately Fast Locomotion over Unperceived Rough Terrain. *Dynamic Walking 2013*, Carnegie Mellon University, Pittsburgh, USA, JUNE 10-13, 2013

Mostafa Ajallooeian, Sébastien Gay, Alexandre Tuleu, Alexander Sproewitz and Auke Ijspeert, Modular Control of Limit Cycle Locomotion over Unperceived Rough Terrain, Accepted for *IEEE International Conference on Intelligent Robots and Systems(IROS), 2013*, Tokyo, Japan, November 3-7 2012

Sébastien Gay, Auke Ijspeert and José Santos-Victor, Predictive Gaze Stabilization During Periodic Locomotion Based on Adaptive Frequency Oscillators, *IEEE International Conference on Robotics and Automation (ICRA), 2012*, pp.271-278, St Paul, USA, 14-18 May 2012

Sébastien Gay, Auke Ijspeert and José Santos-Victor, Robot Head Stabilization During Periodic Locomotion Using Adaptive Dynamical Systems, *International Symposium on Adaptive Motion of Animals and Machines, 2011*

Sarah Dégallier Rochat, Ludovic Righetti, Sébastien Gay and Auke Ijspeert. Towards Simple Control for Complex, Autonomous Robotic Applications: Combining Discrete and Rhythmic Motor Primitives, *Autonomous Robots , 2011* vol. 31, num. 2, p. 155-181.

Sébastien Gay, Sarah Dégallier Rochat, Auke Ijspeert and José Santos-Victor. Integration of vision and central pattern generator based locomotion for path planning of a non-holonomic crawling humanoid robot. *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2010*, Taipei, Taiwan, 18-22 October

Qingqiong Deng, Xiaopeng Zhang, Sebastien Gay and Xiangdong Lei, Continuous LOD Model of Coniferous Foliage, *The International Journal of Virtual Reality, 2007*, 6(4):77-84